Computer Science and Artificial Intelligence Laboratory

Technical Report

MIT-CSAIL-TR-2010-056 December 3, 2010

Verification of Semantic Commutativity Conditions and Inverse Operations on Linked Data Structures

Deokhwan Kim and Martin C. Rinard

massachusetts institute of technology, cambridge, ma 02139 usa — www.csail.mit.edu

Verification of Semantic Commutativity Conditions and Inverse Operations on Linked Data Structures

Deokhwan Kim Martin C. Rinard

Massachusetts Institute of Technology
{dkim,rinard}@csail.mit.edu

Abstract

Commuting operations play a critical role in many parallel computing systems. We present a new technique for verifying *commutativity conditions*, which are logical formulas that characterize when operations commute. Because our technique reasons with the abstract state of verified linked data structure implementations, it can verify commuting operations that produce semantically equivalent (but not identical) data structure states in different execution orders. We have used this technique to verify sound and complete commutativity conditions for all pairs of operations on a collection of linked data structure implementations, including data structures that export a set interface (ListSet and HashSet) as well as data structures that export a map interface (AssociationList, HashTable, and ArrayList). This effort involved the specification and verification of 765 commutativity conditions.

Many speculative parallel systems need to undo the effects of speculatively executed operations. *Inverse operations*, which undo these effects, are often more efficient than alternate approaches (such as saving and restoring data structure state). We present a new technique for verifying such inverse operations. We have specified and verified, for all of our linked data structure implementations, an inverse operation for every operation that changes the data structure state.

Together, the commutativity conditions and inverse operations provide a key resource that language designers and system developers can draw on to build parallel languages and systems with strong correctness guarantees.

1. Introduction

Commuting operations on shared data structures (operations that produce the same result regardless of the order in which they execute) play a central role in many parallel computing systems:

- **Parallelizing Compilers:** If a compiler can statically detect that all operations in a given computation commute, it can generate parallel code for that computation [12].
- **Deterministic Parallel Languages:** Including support for commuting operations in deterministic parallel languages increases the expressive power of the language while preserving guaranteed deterministic parallel execution [2, 13].
- **Transaction Monitors:** If a transaction monitor can detect that operations within parallel transactions commute, it can use efficient locking algorithms that allow commuting operations from different transactions to interleave [7, 16]. Because such locking algorithms place fewer constraints on the execution order, they increase the amount of exploitable parallelism.
- **Irregular Parallel Computations:** Exploiting commuting operations has been shown to be critical for obtaining good parallel performance in irregular parallel computations that manipulate linked data structures [10, 11]. The reason is essentially the same as for efficient transaction monitors — it enables the use of efficient synchronization algorithms for atomic transactions that execute multiple (potentially commuting) operations on shared objects. For similar reasons, exploiting commuting operations has also been shown to be essential for obtaining good parallel performance for the SPEC benchmarks [4].

Despite the importance of commuting operations, there has been relatively little research in automatically analyzing or verifying the conditions under which operations commute. Indeed, the deterministic parallel language, transaction monitor, and irregular parallel computation systems cited above all rely on the developer to identify commuting operations, with no way to determine whether the operations do, in fact, commute or not. A mistake in identifying commuting operations invalidates both the principles upon which the systems operate and the correctness guarantees that they claim to provide.

1.1 Previous Research

Commutativity analysis [12] uses static program analysis to find operations that produce identical concrete object states in all execution orders. But this approach is inadequate for linked data structures — consider, for example, a linked list that implements a set interface. Operations that insert elements into this data structure commute at the semantic level — all insertion orders produce the same abstract set of elements. But they do not commute at the concrete implementation level — different insertion orders (even though they produce the same set) produce different linked lists. Any commutativity analysis that reasons at the concrete implementation level (as opposed to the abstract semantic level) would therefore conservatively conclude that such operations do not commute.

Another approach uses *random interpretation* [9] to detect commuting operations [1]. This technique uses a variant of concolic testing to explore all control flow paths from different execution orders, with affine join operations combining states from different control flow paths to avoid exponential blowup in the number of analyzed states. Instead of directly comparing states from different execution orders, the technique reasons about the return values of the functions that the program uses to observe the different states. Because effective affine join operations do not currently exist for linked data structures, this approach does not detect commuting operations on linked data structures.

Both of these techniques are designed only to find operations that commute in all possible object states and for all possible parameter values. But some operations commute only under certain conditions. Consider, for example, an operation that removes a key, value pair from a hash table and an operation that looks up a given key in the hash table. These operations commute only if the two keys are different. Recognizing and exploiting such *commutativity conditions* is essential to obtaining good parallel performance for many irregular computations [10, 11] — these computations use the commutativity conditions to dynamically recognize and exploit commuting operations whose commutativity properties they cannot statically resolve.

1.2 Semantic Commutativity Analysis

This paper presents a new analysis for verifying the specific conditions under which operations on linked data structures seman-

tically commute. Instead of reasoning directly about the concrete data structure state, this analysis builds on the availability of fully verified linked data structure implementations [17, 18] to reason at the higher semantic level of the (verified) abstract data structure state. The analysis is therefore able to detect operations that commute at the semantic level even though they may produce different concrete data structure states. The verified commutativity conditions are both *sound* (conceptually, if the conditions hold, the operations produce the same abstract data structure state regardless of the order in which they execute, see Section 4) and *complete* (conceptually, if the conditions do not hold, then different execution orders produce different abstract data structure states, see Section 4).

We have verified sound and complete commutativity conditions for all pairs of operations from a variety of linked data structure implementations:

- **Sets:** ListSet and HashSet both implement a set interface. ListSet uses a singly-linked list; HashSet uses a hash table.
- **Maps:** AssociationList, HashTable, and ArrayList all implement a map interface. AssociationList uses a singly-linked list of key, value pairs; HashTable implements a separately-chained hash table — an array contains linked lists of key, value pairs with a hash function mapping keys to linked lists via the array. ArrayList maps integers to objects and is optimized for storing maps from a dense subset of the integers starting at 0.
- **Accumulator:** Accumulator maintains a value that clients can increase and read.

Altogether, we specified and verified a total of 765 commutativity conditions (216 from ListSet and HashSet, 294 from AssociationList and HashTable, 243 from ArrayList, and 12 from Accumulator).

The well-known difficulty of reasoning about semantic properties of linked data structures [14] has limited the range of available results in this area. These verified commutativity conditions therefore provide a solid foundation for the use of these linked data structures in a range of parallel programs and systems.

1.3 Inverse Operations

In one of our usage scenarios, the system uses the commutativity conditions to dynamically detect speculatively executed operations that do not commute with previously executed operations [10, 11]. In this case, the system must roll the data structure back to the abstract semantic state before the operations executed, then continue from this restored state. Executing *inverse operations* that undo the effect of executed operations can be substantially more efficient than alternate approaches (such as pessimistically saving the data structure state before operations execute, then restoring the state to roll back the effect of the operations). Note that even though the restored abstract semantic state is the same, the underlying concrete states may differ. For example, the inverse of an operation that removes an element from a set implemented as a linked list inserts the removed element back into the list. Even though the reinserted element may appear in a different position in the list, the restored abstract set is the same as the original set. We have developed an analysis that is capable of verifying semantic inverse operations and used this analysis to verify inverses for operations that modify the abstract semantic state of our linked data structures.

The need to undo the effects of executed operations occurs pervasively throughout computer systems, from classical database transaction processing systems [8] to systems that recover from security breaches. In addition to the specific motivating use described above, the verified inverse operations may therefore find broader applicability in a variety of contexts in which it is desirable to efficiently undo data structure state changes.

1.4 Jahob

We use the Jahob program specification and verification system to specify and verify the data structure implementations, commutativity conditions, and inverse operations. Jahob enables developers to write higher-order logic specifications for Java programs [18]. It also enables developers to guide proofs of complex program properties by using the Jahob integrated proof language to resolve key choice points in these proofs. Once these choice points have been resolved, Jahob uses *integrated reasoning* to invoke a variety of powerful reasoning systems (such as first-order provers, SMT provers, MONA, and the BAPA decision procedure) to discharge the resulting automatically generated verification conditions [17, 18].

Our commutativity condition and inverse operation verification system generates stylized Jahob methods whose verification establishes the validity of the corresponding commutativity conditions and inverse operations. Of the 1530 commutativity testing methods, all but 57 verify as generated with no need for developer intervention. We augmented the remaining methods to include the Jahob proof statements required to enable Jahob to complete the verification. All of the inverse testing methods verify as generated.

1.5 Contributions

This paper makes the following contributions:

- **Semantic Commutativity Analysis:** It presents a new commutativity analysis technique that verifies sound and complete semantic commutativity conditions for linked data structures. Because this analysis reasons about the abstract semantic state (as opposed to the concrete implementation state), it can verify semantic commutativity conditions that are inherently beyond the reach of previously proposed approaches.

  To the best of our knowledge, this analysis is the first to verify semantic commutativity conditions for linked data structures.
- **Commutativity Conditions:** It presents verified sound and complete commutativity conditions for a variety of linked data structures. In this paper all of these commutativity conditions are provided by the developer and verified by our implemented system.

  To the best of our knowledge, these are the first fully verified semantic commutativity conditions for linked data structures.
- **Semantic Inverse Analysis:** It presents a new analysis for verifying inverse operations that undo the effect of previously executed operations on linked data structures. Because the analysis reasons about the abstract data structure state, it can verify semantic inverses that correctly restore the abstract data structure state even though they may produce different concrete states.

  To the best of our knowledge, this analysis is the first to verify semantic inverse operations for linked data structures.
- **Inverse Operations:** It presents verified inverses for operations that update the data structure state. Systems can use these operations to efficiently roll back speculatively executed data structure operations.

  To the best of our knowledge, these are the first fully verified inverse operations for linked data structures.
- **Experience:** It discusses our experience using the Jahob [3, 17, 18] program specification and verification system to specify and verify the commutativity conditions and inverse operations.

2. Example

Figure 1 presents the Jahob interface for the HashSet class, which uses a separately-chained hash table [5] to implement a

```
1  public class HashSet {
2      /*: public ghost specvar init :: "bool" = "False"; */
3      /*: public ghost specvar contents :: "obj set" = "{}"; */
4      /*: public specvar size :: "int"; */
5      private Node[] table;
6      private int _size;
7      public HashSet()
8      /*: modifies "init", "contents", "size"
9          ensures "init & contents = {} & size = 0" */ { ... }
10     public boolean add(Object v)
11     /*: requires "init & v ~= null"
12         modifies "contents", "size"
13         ensures "(v ~: old contents --> contents = old contents Un {v} & size = old size + 1 & result) &
14                  (v : old contents --> contents = old contents & size = old size & ~result)" */ { ... }
15     public boolean contains(Object v)
16     /*: requires "init & v ~= null"
17         ensures "result = (v : contents)" */ { ... }
18     public boolean remove(Object v)
19     /*: requires "init & v ~= null"
20         modifies "contents", "size"
21         ensures "(v : old contents --> contents = old contents - {v} & size = old size - 1 & result) &
22                  (v ~: old contents --> contents = old contents & size = old size & ~result)" */ { ... }
23     public int size()
24     /*: requires "init"
25         ensures "result = size" */ { ... }
26 }
```

Figure 1. The Jahob HashSet Specification

set interface. The HashSet class, like all of our data structures, is written in Java augmented with specifications written in the Jahob higher-order logic specification language. The interface exports a collection of specified operations. Each operation specification consists of a precondition (the `requires` clause), a postcondition (the `ensures` clause), and a `modifies` clause. These specifications completely capture the desired behavior of the data structure (with the exception of properties involving execution time and/or memory consumption) [17, 18].

Abstract State The interface uses the *abstract state* of the HashSet to specify the behavior of HashSet operations. This state consists of the set `contents` of objects in the HashSet, the `size` of this set, and the flag `init`, which is true if the HashSet has been initialized (see lines 2, 3, and 4 of Figure 1). The specification for the `add(v)` operation, for example, uses this abstract state to specify that, if the HashSet is initialized and the parameter `v` is not null, it adds `v` to the set of objects in the HashSet (see lines 11-14 of Figure 1).

Concrete State and Abstraction Function When the program runs, the HashSet operations manipulate the *concrete state* of the HashSet. The concrete state consists of the array `table`, which contains pointers to linked lists of elements in the HashSet, and the `int _size`, which stores the size of the table (see lines 5 and 6 of Figure 1). An *abstraction function* in the form of Jahob invariants (not shown, but see the complete data structure specifications and implementations available in Appendix B and in the package of source code that accompanies the paper) specifies the relationship between the concrete and abstract states [17, 18]. Like all of the data structures in this paper, we have used the Jahob system to verify that the HashSet correctly implements its interface [17, 18]. This verification, of course, includes the verification of the abstraction function.

Commuting Operations Consider the `add(v1)` and `contains(v2)` operations on a HashSet `s`. These operations commute if and only if `v1` does not equal `v2` or `v1` is already in `s`. Figure 2 presents the two methods that our system automatically generates to verify the soundness and completeness of this commutativity condition. The first method (`contains_add_between_s_40`, line 1 of Figure 2) verifies soundness. The second method (`contains_add_between_c_40`, line 14 of Figure 2) verifies completeness. The methods are written in a subset of Java with Jahob [18] annotations.

Verifying Commutativity Condition Soundness The basic approach for verifying soundness is to generate code that executes the `add(v1)` and `contains(v2)` operations in both execution orders on equivalent HashSets (HashSets with the same abstract state). The code then checks that, if the commutativity condition is true, then both execution orders produce the same return values and final abstract HashSet states.

The `requires` clause (lines 2 and 3 of Figure 2) ensures that the method starts with two HashSets (`sa` and `sb`) that have identical abstract states. The method first applies the `sa.contains(v1)` and `sa.add(v2)` operations to one of the HashSets (`sa`), using a Jahob `assume` statement (line 8 of Figure 2) to instruct Jahob to assume the commutativity condition (in this case `v1 = v2 | r1a`). The method next executes the two operations in the reverse order on the second HashSet `sb` (lines 10 and 11 of Figure 2). The `assert` clause at the end of the method (line 12 of Figure 2) checks that the return values are the same in both execution orders and that the two HashSets have the same abstract states at the end of the method.

In this example the commutativity condition works with the *between state* that is available after the first operation executes but

```
static void contains_add_between_s_40(HashSet sa, HashSet sb, Object v1, Object v2)
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null & v2 ~= null &
             sa..contents = sb..contents & sa..size = sb..size"
    modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
    ensures "True" */
{
    boolean r1a = sa.contains(v1);
    /*: assume "v1 ~= v2 | r1a" */
    sa.add(v2);

    sb.add(v2);
    boolean r1b = sb.contains(v1);

    /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
}

static void contains_add_between_c_40(HashSet sa, HashSet sb, Object v1, Object v2)
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null & v2 ~= null &
             sa..contents = sb..contents & sa..size = sb..size"
    modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
    ensures "True" */
{
    boolean r1a = sa.contains(v1);
    /*: assume "~(v1 ~= v2 | r1a)" */
    sa.add(v2);

    sb.add(v2);
    boolean r1b = sb.contains(v1);

    /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
}
```

Figure 2. HashSet Commutativity Testing Methods for Between Commutativity Condition for `contains(v1)` and `add(v2)`

before the second operation executes. A system would use such a *between condition* just before executing the `add(v2)` operation to dynamically check if this operation commutes with a previously executed `contains(v1)` operation. In addition to these between commutativity conditions, we also verify *before conditions* (which may be used to determine if two operations that have yet to execute will commute when they execute) and *after conditions* (which may be used to trigger rollbacks when already executed operations do not commute [10, 11]). Note that we assume that, in parallel execution contexts, the data structure operations execute atomically using, for example, a mechanism such as locking [10] or transactional memory [15].

The last step is to invoke the Jahob program verification system to verify the generated method. If Jahob verifies the method, it has verified that if the commutativity condition holds, then the operations commute.

Verifying Commutativity Condition Completeness The `contains_add_between_c_40` method, which checks completeness, uses a similar pattern except it negates both the commutativity condition and the assertion at the end of the generated method. If Jahob verifies the method, it has verified that if the commutativity condition does not hold, then the operations produce different return values or different abstract data structure states when they execute in different orders.

For our set of linked data structures, Jahob is able to verify all but 57 of the 1530 generated commutativity testing methods as generated. We used the integrated Jahob proof language [18] to augment the remaining 57 methods with proof commands that enabled Jahob to resolve key choice points in the verification proof and verify the method.

3. Commutativity and Inverse Testing Methods

The commutativity testing method generator takes as input the data structure interface and, for each pair of data structure operations, developer-specified before, between, and after commutativity testing conditions. It produces as output the commutativity testing methods. It then presents each method to the Jahob program verification system [17, 18]. If the method verifies, the system has verified the corresponding commutativity condition. If it does not verify, either the commutativity condition is not sound or complete or Jahob is not capable of verifying the soundness and completeness without additional developer assistance. The developer then, as appropriate, either modifies the commutativity condition or augments the generated commutativity testing methods with additional proof commands written in the Jahob proof language [18].

3.1 Completeness Commutativity Testing Template

Figure 3 presents the template that the generator uses to produce the completeness commutativity testing method. The generation process simply iterates over all commutativity testing conditions (and corresponding pairs of operations in the data structure interface), filling in the template parameters as appropriate. In Figure 3 all parameters appear in *italic font*.

The name of the commutativity testing method contains the names of the two operations, a field that specifies whether the method tests a before, between, or after commutativity condition, the tag `c` (which identifies the method as a completeness testing method), and a numerical identifier *id*. The method takes as parameters two data structures (*sa* and *sb*) and the parameters of the two data structure operations. The requires clause ensures the data structures are distinct but have identical abstract states.

The generated method uses Jahob `assume` statements to instruct Jahob to assume that the preconditions of the operations hold in the first execution order. If the preconditions do not involve the state

of the data structure (as in our example in Figure 2), the generator moves the preconditions up into the requires clause. The generator also uses an assume statement to insert the negation of the commutativity condition (recall that the template is a completeness template and therefore includes the negation of the condition) in the appropriate place in the generated method. The template identifies the insertion points for all three kinds of commutativity conditions (before, between, and after). A generated method, of course, contains a commutativity condition at only one of these points.

The method next contains the operations in the reverse order, with `assume` statements instructing Jahob to assume that the preconditions of the operations hold. Once again, if the preconditions do not depend on the data structure state, the generator places them in the requires clause of the method, not before the operation invocations as in the template.

The method ends with the final assertion (which Jahob must prove) that either one of the corresponding return values or the final abstract states are different in the two different execution orders.

As is appropriate for a completeness testing method, this structure forces Jahob to prove that if the operation preconditions and the negation of the commutativity condition holds in the first execution order, then either one of the operation preconditions is violated in the reverse execution order or the final assertion holds.

3.2 Soundness Commutativity Testing Template

The soundness commutativity testing template has the same basic structure as the completeness template, with the exception that 1) it inserts the commutativity testing condition (not its negation), 2) it omits the `assume` statements for the operation preconditions in the second execution order, and 3) the final assertion forces Jahob to prove that the return values and final abstract states are the same in both execution orders.

As is appropriate for a soundness testing method, this structure forces Jahob to prove that if the operation preconditions and commutativity condition holds in the first execution order, then the operation preconditions hold in the reverse execution order and the return values and final abstract states are the same.

3.3 Inverse Testing Methods

The inverse testing method generator takes as input the data structure interface and a developer-specified set of inverse operation pairs. It produces as output the inverse testing methods and feeds each method to the Jahob program verification system [18]. As for the commutativity testing methods, the developer may, if necessary, augment the inverse testing methods with additional Jahob proof commands.

Figure 4 presents the template that the generator uses to produce the inverse testing methods. The generation process simply iterates over all of the specified inverses, filling in the template parameters (in *italic font*) as appropriate. The final Jahob `assert` statement requires Jahob to prove that the final abstract state (after the application of the inverse operation) is the same as the initial abstract state from the start of the method. Jahob must also prove the precondition of the inverse operation.

```
static void method1_method2_(before | between | after)_c_id
              (sa_decl, sb_decl, argv1_decls, argv2_decls)
/*: requires "sa ~= null & sb ~= null & sa ~= sb &
              sa_abstract_state = sb_abstract_state"
    modifies "sa_frame_condition", "sb_frame_condition"
    ensures "True" */
{
    [/*: assume "~(before_commutativity_condition)" */]
    /*: assume "method1_precondition" */
    r1a_type r1a = sa.method1(argv1);
    [/*: assume "~(between_commutativity_condition)" */]
    /*: assume "method2_precondition" */
    r2a_type r2a = sa.method2(argv2);
    [/*: assume "~(after_commutativity_condition)" */]

    /*: assume "method2_precondition" */
    r2b_type r2b = sb.method2(argv2);
    /*: assume "method1_precondition" */
    r1b_type r1b = sb.method1(argv1);

    /*: assert "~(r1a = r1b & r2a = r2b &
                  sa_abstract_state = sb_abstract_state)" */
}
```

Figure 3. Template for Commutativity Testing Methods

```
static void method_id(s_decl, argv_decls)
/*: requires "s ~= null & method_precondition"
    modifies "s_frame_condition"
    ensures "True" */
{
    r_type r = s.method(argv);
    execute_inverse_operation();

    /*: assert "s_abstract_state = s_initial_abstract_state" */
}
```

Figure 4. Template for Inverse Testing Methods

4. Formal Treatment

We assume a set $s \in S$ of concrete states and a corresponding set $\widehat{s} \in \widehat{S}$ of abstract states. We also assume that the data structure defines an abstraction function $\alpha : S \rightarrow \widehat{S}$.

The commutativity and inverse testing methods work with logical formulas written in the higher-order logic Jahob specification language [18]. For our data structures, the specifications, commutativity conditions, commutativity testing methods, and inverse testing methods require only first-order logic.

Given an operation $m(v)$ on a given data structure, $\mathrm{pre}(m(v))$ denotes the precondition of the method m from the data structure specification. The precondition is a logical formula written in the Jahob specification language [18]. It is expressed in the name space of the caller (i.e., with the formal parameter from the definition of m replaced by the actual parameter v from the caller). We write $\alpha(s) \models \mathrm{pre}(m(v))$ if the precondition is true in the abstract state $\alpha(s)$.

We write $\langle s', r \rangle = s.m(v)$ if executing the operation $m(v)$ in state s produces return value r and new state s'. Given a starting state s and two operations $m_1(v_1)$ and $m_2(v_2)$, we are interested in the following states and return values (see Figure 5):

- $\langle s_{\underline{1};\underline{2}}, r_{\underline{1};\underline{2}} \rangle = s.m_1(v_1)$: the intermediate state $s_{\underline{1};\underline{2}}$ and return value $r_{\underline{1};\underline{2}}$ that results from executing $m_1(v_1)$ in the original state s.
- $\langle s_{1;\underline{2}}, r_{1;\underline{2}} \rangle = s_{\underline{1};\underline{2}}.m_2(v_2)$: the final state $s_{1;\underline{2}}$ and and return value $r_{1;\underline{2}}$ that results from executing $m_2(v_2)$ in the intermediate state $s_{\underline{1};2}$.
- $\langle s_{\underline{2};1}, r_{\underline{2};1} \rangle = s.m_2(v_2)$: the intermediate state $s_{\underline{2};1}$ and return value $r_{\underline{2};1}$ that results from executing $m_2(v_2)$ in the original state s.
- $\langle s_{2;\underline{1}}, r_{2;\underline{1}} \rangle = s_{\underline{2};1}.m_1(v_1)$: the final state $s_{2;\underline{1}}$ and and return value $r_{2;\underline{1}}$ that results from executing $m_1(v_1)$ in the intermediate state $s_{\underline{2};1}$.

A commutativity condition ϕ is a logical formula written in the Jahob specification language [18]. In general, the free variables of ϕ

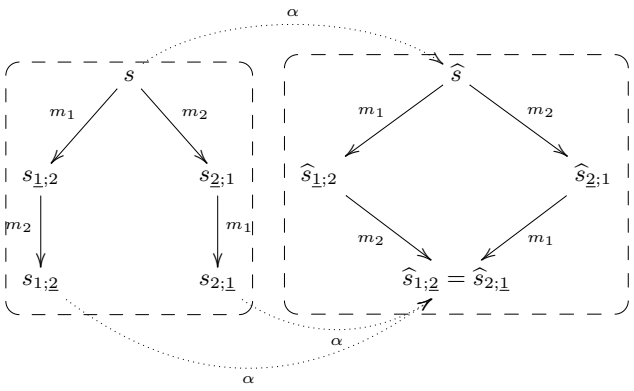

Figure 5. Execution on Concrete States and Abstract States

can include the arguments v_1 and v_2, the return values $r_{\underline{1},2}$ and $r_{1,\underline{2}}$, and abstract specification variables that denote various components of the three abstract states $\alpha(s)$, $\alpha(s_{\underline{1};2})$, and $\alpha(s_{1;\underline{2}})$. We write $(\langle s_{\underline{1};2}, r_{\underline{1};2}\rangle = s.m_1(v_1); \langle s_{1;\underline{2}}, r_{1;\underline{2}}\rangle = s_{\underline{1};2}.m_2(v_2)) \models \phi$ if the commutativity condition ϕ is satisfied when the operations execute in the order $m_1(v_1); m_2(v_2)$ (first $m_1(v_1)$, then $m_2(v_2)$).

Given a commutativity condition ϕ for two operations $m_1(v_1)$ and $m_2(v_2)$, the verification of the soundness commutativity testing method for these two operations establishes (by the construction of this method) the following property:

Property 1 (soundness)**.** *If* $\alpha(s) \models \text{pre}(m_1(v_1))$, $\alpha(s_{\underline{1};2}) \models \text{pre}(m_2(v_2))$, *and* $(\langle s_{\underline{1};2}, r_{\underline{1};2}\rangle = s.m_1(v_1); \langle s_{1;\underline{2}}, r_{1;\underline{2}}\rangle = s_{\underline{1};2}.m_2(v_2)) \models \phi$ *then* $\alpha(s) \models \text{pre}(m_2(v_2))$, $\alpha(s_{\underline{2};1}) \models \text{pre}(m_1(v_1))$, $r_{\underline{1};2} = r_{2;\underline{1}}$, $r_{1;\underline{2}} = r_{\underline{2};1}$, *and* $\alpha(s_{1;\underline{2}}) = \alpha(s_{2;\underline{1}})$.

Given a commutativity condition ϕ for two operations $m_1(v_1)$ and $m_2(v_2)$, the verification of the completeness commutativity testing method for these two operations establishes (by the construction of this method) the following property:

Property 2 (completeness)**.** *If* $\alpha(s) \models \text{pre}(m_1(v_1))$, $\alpha(s_{\underline{1};2}) \models \text{pre}(m_2(v_2))$, *and* $(\langle s_{\underline{1};2}, r_{\underline{1};2}\rangle = s.m_1(v_1); \langle s_{1;\underline{2}}, r_{1;\underline{2}}\rangle = s_{\underline{1};2}.m_2(v_2)) \models \phi$ *then* $\alpha(s) \models \text{pre}(m_2(v_2))$, $\alpha(s_{\underline{2};1}) \models \text{pre}(m_1(v_1))$, $r_{\underline{1};2} \neq r_{2;\underline{1}}$, $r_{1;\underline{2}} \neq r_{\underline{2};1}$, *or* $\alpha(s_{1;\underline{2}}) \neq \alpha(s_{2;\underline{1}})$.

Given an operation $m_1(v_1)$ with inverse operation $m_2(v_2)$, the verification of the inverse testing method for these two operations establishes (by the construction of this method) the following property:

Property 3 (inverse)**.** *If* $\alpha(s) \models \text{pre}(m_1(v_1))$ *then* $\alpha(s_{\underline{1};2}) \models \text{pre}(m_2(v_2))$ *and* $\alpha(s) = \alpha(s_{1;\underline{2}})$.

5. Experimental Results

We next discuss the commutativity conditions, inverse operations, and verification process for our set of data structures. We first discuss the operations that each data structure exports. The source code for all of the data structures (including both specification and implementation) as well as the commutativity and inverse testing methods (which contain all of the commutativity conditions and Jahob proof constructs required to enable Jahob to verify the methods) is available in Appendix B and in the ancillary file that accompanies the paper.

The Accumulator implements a counter with two operations:

- `increase(v)`: Adds the number `v` to the counter.
- `read()`: Returns the value in the counter.

HashSet and ListSet implement a set of elements with the following operations. Because they implement the same specification, they have the same commutativity conditions.

- `add(v)`: Adds the element `v` to the set of elements in the data structure. Returns false if the element was already present and true otherwise.
- `contains(v)`: Returns true if the element `v` is in the set and false otherwise.
- `remove(v)`: Removes the element `v` from the set. Returns true if `v` was included in the set and false otherwise.
- `size()`: Returns the number of elements in the set.

HashTable and AssociationList implement a map from keys to values with the following operations. Because they implement the same specification, they have the same commutativity conditions.

- `containsKey(k)`: Returns true if there exists a value `v` for the key `k` in the map.
- `get(k)`: Returns the value `v` for the key `k`, or `null` if `k` is not mapped.
- `put(k, v)`: Maps the key `k` to the value `v`. Returns the previous value for the key `k`, or `null` if `k` was not mapped.
- `remove(k)`: Removes the mapping for the key `k`. Returns the value that the key `k` was mapped to, or `null` if the the data structure did not have a mapping for the key `k`.
- `size()`: Returns the number of key, value pairs in the data structure.

ArrayList implements a map from the integers to objects with the following operations:

- `add_at(i, v)`: Pushes all objects with indices greater than or equal to `i` up one position to create an empty position at index `i`, then inserts the object `v` into that position.
- `get(i)`: Returns the object at index `i`.
- `indexOf(v)`: Returns the index of the first occurrence of the object `v` or -1 if the object `v` is not in the map.
- `lastIndexOf(v)`: Returns the index of the last occurrence of the object `v` or -1 if the object `v` is not in the map.
- `remove_at(i)`: Removes the element at the specified index `i`, then slides all objects above `i` down one position to fill the newly empty position at index `i`.
- `set(i, v)`: Replaces the object at the index `i` with the object `v`. Returns the replaced object previously at index `i` or `null` if there was no such object.
- `size()`: Returns the number of elements in the map.

5.1 Commutativity Conditions

Tables 1 through 7 present the commutativity conditions for selected illustrative pairs of operations from our set of linked data structures. The complete set of all 765 commutativity conditions is available in Appendix A.

The first and second columns in each table identify the pair of operations. The third column presents the commutativity conditions in terms of the arguments, return values, and abstract data structure states. These commutativity conditions are suitable for static analyses that reason about the commutativity conditions at the level of the abstract states. The fourth column translates any abstract state queries (typically set membership operations) into operations on that can be invoked on the concrete data structure. These commutativity conditions are suitable for dynamically checking the commutativity conditions when the program runs.

Methods		Commutativity Condition	
$s_1.\texttt{increase}(v_1)$	$s_2.\texttt{increase}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{read}()$	$v_1 = 0$	$v_1 = 0$
$r_1 = s_1.\texttt{read}()$	$s_2.\texttt{increase}(v_2)$	$v_2 = 0$	$v_2 = 0$
	$r_2 = s_2.\texttt{read}()$	$true$	$true$

Table 1. Commutativity Conditions on Accumulator

Methods		Commutativity Condition	
$s_1.\texttt{add}(v_1)$	$s_2.\texttt{add}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{contains}(v_2)$	$v_1 \neq v_2 \vee v_1 \in s_1$	$v_1 \neq v_2 \vee s_1.\texttt{contains}(v_1) = true$
	$s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2$	$v_1 \neq v_2$
$r_1 = s_1.\texttt{contains}(v_1)$	$s_2.\texttt{add}(v_2)$	$v_1 \neq v_2 \vee v_1 \in s_1$	$v_1 \neq v_2 \vee s_1.\texttt{contains}(v_1) = true$
	$r_2 = s_2.\texttt{contains}(v_2)$	$true$	$true$
	$s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2 \vee v_1 \notin s_1$	$v_1 \neq v_2 \vee s_1.\texttt{contains}(v_1) = false$
$s_1.\texttt{remove}(v_1)$	$s_2.\texttt{add}(v_2)$	$v_1 \neq v_2$	$v_1 \neq v_2$
	$r_2 = s_2.\texttt{contains}(v_2)$	$v_1 \neq v_2 \vee v_1 \notin s_1$	$v_1 \neq v_2 \vee s_1.\texttt{contains}(v_1) = false$
	$s_2.\texttt{remove}(v_2)$	$true$	$true$

Table 2. Before Commutativity Conditions on ListSet and HashSet

Methods		Commutativity Condition	
$s_1.\texttt{add}(v_1)$	$s_2.\texttt{add}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{contains}(v_2)$	$v_1 \neq v_2 \vee v_1 \in s_1$	$v_1 \neq v_2 \vee s_1.\texttt{contains}(v_1) = true$
	$s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2$	$v_1 \neq v_2$
$r_1 = s_1.\texttt{contains}(v_1)$	$s_2.\texttt{add}(v_2)$	$v_1 \neq v_2 \vee r_1 = true$	$v_1 \neq v_2 \vee r_1 = true$
	$r_2 = s_2.\texttt{contains}(v_2)$	$true$	$true$
	$s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2 \vee r_1 = false$	$v_1 \neq v_2 \vee r_1 = false$
$s_1.\texttt{remove}(v_1)$	$s_2.\texttt{add}(v_2)$	$v_1 \neq v_2$	$v_1 \neq v_2$
	$r_2 = s_2.\texttt{contains}(v_2)$	$v_1 \neq v_2 \vee v_1 \notin s_1$	$v_1 \neq v_2 \vee s_1.\texttt{contains}(v_1) = false$
	$s_2.\texttt{remove}(v_2)$	$true$	$true$

Table 3. Between Commutativity Conditions on ListSet and HashSet

Methods		Commutativity Condition	
$r_1 = s_1.\texttt{get}(k_1)$	$r_2 = s_2.\texttt{get}(k_2)$	$true$	$true$
	$s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2 \vee \langle k_1, v_2 \rangle \in s_1$	$k_1 \neq k_2 \vee s_1.\texttt{get}(k_1) = v_2$
	$s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, \_ \rangle \notin s_1$	$k_1 \neq k_2 \vee s_1.\texttt{containsKey}(k_1) = false$
$s_1.\texttt{put}(k_1, v_1)$	$r_2 = s_2.\texttt{get}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, v_1 \rangle \in s_1$	$k_1 \neq k_2 \vee s_1.\texttt{get}(k_1) = v_1$
	$s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2 \vee v_1 = v_2$	$k_1 \neq k_2 \vee v_1 = v_2$
	$s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2$	$k_1 \neq k_2$
$s_1.\texttt{remove}(k_1)$	$r_2 = s_2.\texttt{get}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, \_ \rangle \notin s_1$	$k_1 \neq k_2 \vee s_1.\texttt{containsKey}(k_1) = false$
	$s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2$	$k_1 \neq k_2$
	$s_2.\texttt{remove}(k_2)$	$true$	$true$

Table 4. Before Commutativity Conditions on AssociationList and HashTable

Methods		Commutativity Condition	
$r_1 = s_1.\texttt{get}(k_1)$	$r_2 = s_2.\texttt{get}(k_2)$	$true$	$true$
	$s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2 \vee r_1 = v_2$	$k_1 \neq k_2 \vee r_1 = v_2$
	$s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2 \vee r_1 = null$	$k_1 \neq k_2 \vee r_1 = null$
$s_1.\texttt{put}(k_1, v_1)$	$r_2 = s_2.\texttt{get}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, v_1 \rangle \in s_1$	$k_1 \neq k_2 \vee s_1.\texttt{get}(k_1) = v_1$
	$s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2 \vee v_1 = v_2$	$k_1 \neq k_2 \vee v_1 = v_2$
	$s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2$	$k_1 \neq k_2$
$s_1.\texttt{remove}(k_1)$	$r_2 = s_2.\texttt{get}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, \_ \rangle \notin s_1$	$k_1 \neq k_2 \vee s_1.\texttt{containsKey}(k_1) = false$
	$s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2$	$k_1 \neq k_2$
	$s_2.\texttt{remove}(k_2)$	$true$	$true$

Table 5. After Commutativity Conditions on AssociationList and HashTable

Methods		Commutativity Condition	
s_1.add_at(i_1, v_1)	s_2.add_at(i_2, v_2)	$(i_1 < i_2 \leq \lvert s_2 \rvert - 1 \wedge s_2[i_2] = v_2) \vee$ $(i_1 = i_2 \wedge v_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_2[i_1 - 1] = v_1)$	$(i_1 < i_2 \leq s_2$.size$() - 1 \wedge s_2$.get$(i_2) = v_2) \vee$ $(i_1 = i_2 \wedge v_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_2$.get$(i_1 - 1) = v_1)$
	$r_2 = s_2$.indexOf(v_2)	$\neg(\exists i : s_2[i] = v_2) \vee$ $(\exists i < i_1 : s_2[i] = v_2) \vee$ $(\neg(\exists i < i_1 : s_2[i] = v_2) \wedge s_2[i_1] = v_2 \wedge s_2[i_1 + 1] = v_2)$	s_2.indexOf$(v_2) < 0 \vee$ $0 \leq s_2$.indexOf$(v_2) < i_1 \vee$ $(s_2$.indexOf$(v_2) = i_1 \wedge s_2$.get$(i_1 + 1) = v_2)$
	s_2.remove_at(i_2)	$(i_1 < i_2 < \lvert s_2 \rvert - 1 \wedge s_2[i_2] = s_2[i_2 + 1]) \vee$ $(\lvert s_2 \rvert - 2 \geq i_1 = i_2 \wedge s_2[i_1 + 1] = v_1) \vee$ $(\lvert s_2 \rvert - 2 \geq i_1 > i_2 \wedge s_2[i_1 + 1] = v_1)$	$(i_1 < i_2 < s_2$.size$() - 1 \wedge s_2$.get$(i_2) = s_2$.get$(i_2 + 1)) \vee$ $(s_2$.size$() - 2 \geq i_1 = i_2 \wedge s_2$.get$(i_1 + 1) = v_1) \vee$ $(s_2$.size$() - 2 \geq i_1 > i_2 \wedge s_2$.get$(i_1 + 1) = v_1)$
$r_1 = s_1$.indexOf(v_1)	s_2.add_at(i_2, v_2)	$(r_1 < 0 \wedge v_1 \neq v_2) \vee$ $0 \leq r_1 < i_2 \vee$ $(r_1 = i_2 \wedge v_1 = v_2)$	$(r_1 < 0 \wedge v_1 \neq v_2) \vee$ $0 \leq r_1 < i_2 \vee$ $(r_1 = i_2 \wedge v_1 = v_2)$
	$r_2 = s_2$.indexOf(v_2)	$true$	$true$
	s_2.remove_at(i_2)	$r_1 < 0 \vee$ $0 \leq r_1 < i_2 \vee$ $(r_1 = i_2 \wedge i_2 < \lvert s_2 \rvert - 1 \wedge s_2[i_2 + 1] = v_1)$	$r_1 < 0 \vee$ $0 \leq r_1 < i_2 \vee$ $(r_1 = i_2 \wedge i_2 < s_2$.size$() - 1 \wedge s_2$.get$(i_2 + 1) = v_1)$
s_1.remove_at(i_1)	s_2.add_at(i_2, v_2)	$(i_1 < i_2 \wedge s_2[i_2 - 1] = v_2) \vee$ $(i_1 = i_2 \wedge s_1[i_1] = v_2) \vee$ $(i_1 > i_2 \wedge s_2[i_1 - 1] = s_1[i_1])$	$(i_1 < i_2 \wedge s_2$.get$(i_2 - 1) = v_2) \vee$ $(i_1 = i_2 \wedge s_1$.get$(i_1) = v_2) \vee$ $(i_1 > i_2 \wedge s_2$.get$(i_1 - 1) = s_1$.get$(i_1))$
	$r_2 = s_2$.indexOf(v_2)	$(\neg(\exists i : s_2[i] = v_2) \wedge s_1[i_1] \neq v_2) \vee$ $(\exists i < i_1 : s_2[i] = v_2) \vee$ $(\neg(\exists i < i_1 : s_2[i] = v_2) \wedge s_2[i_1] = v_2 \wedge s_1[i_1] = v_2 \wedge i_1 < \lvert s_2 \rvert)$	$(s_2$.indexOf$(v_2) < 0 \wedge s_1$.get$(i_1) \neq v_2) \vee$ $0 \leq s_2$.indexOf$(v_2) < i_1 \vee$ $(s_2$.indexOf$(v_2) = i_1 \wedge s_1$.get$(i_1) = v_2 \wedge i_1 < s_2$.size$())$
	s_2.remove_at(i_2)	$(i_1 < i_2 \wedge s_2[i_2 - 1] = s_2[i_2]) \vee$ $i_1 = i_2 \vee$ $(\lvert s_2 \rvert > i_1 > i_2 \wedge s_1[i_1] = s_2[i_1])$	$(i_1 < i_2 \wedge s_2$.get$(i_2 - 1) = s_2$.get$(i_2)) \vee$ $i_1 = i_2 \vee$ $(s_2$.size$() > i_1 > i_2 \wedge s_1$.get$(i_1) = s_2$.get$(i_1))$

Table 6. Between Commutativity Conditions on ArrayList

Methods		Commutativity Condition	
s_1.add_at(i_1, v_1)	s_2.add_at(i_2, v_2)	$(i_1 < i_2 \leq \lvert s_3 \rvert - 2 \wedge s_3[i_2 + 1] = v_2) \vee$ $(i_1 = i_2 \wedge v_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_3[i_1] = v_1)$	$(i_1 < i_2 \leq s_3$.size$() - 2 \wedge s_3$.get$(i_2 + 1) = v_2) \vee$ $(i_1 = i_2 \wedge v_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_3$.get$(i_1) = v_1)$
	$r_2 = s_2$.indexOf(v_2)	$r_2 < 0 \vee$ $0 \leq r_2 < i_1 \vee$ $(r_2 = i_1 \wedge s_3[i + 1] = v_2)$	$r_2 < 0 \vee$ $0 \leq r_2 < i_1 \vee$ $(r_2 = i_1 \wedge s_3$.get$(i_1 + 1) = v_2)$
	s_2.remove_at(i_2)	$(i_1 < i_2 < \lvert s_3 \rvert \wedge s_2[i_2] = s_3[i_2]) \vee$ $(\lvert s_3 \rvert - 1 \geq i_1 = i_2 \wedge s_3[i_1] = v_1) \vee$ $(\lvert s_3 \rvert - 1 \geq i_1 > i_2 \wedge s_3[i_1] = v_1)$	$(i_1 < i_2 < s_3$.size$() \wedge s_2$.get$(i_2) = s_3$.get$(i_2)) \vee$ $(s_3$.size$() - 1 \geq i_1 = i_2 \wedge s_3$.get$(i_1) = v_1) \vee$ $(s_3$.size$() - 1 \geq i_1 > i_2 \wedge s_3$.get$(i_1) = v_1)$
$r_1 = s_1$.indexOf(v_1)	s_2.add_at(i_2, v_2)	$(r_1 < 0 \wedge v_1 \neq v_2) \vee$ $0 \leq r_1 < i_2 \vee$ $(r_1 = i_2 \wedge v_1 = v_2)$	$(r_1 < 0 \wedge v_1 \neq v_2) \vee$ $0 \leq r_1 < i_2 \vee$ $(r_1 = i_2 \wedge v_1 = v_2)$
	$r_2 = s_2$.indexOf(v_2)	$true$	$true$
	s_2.remove_at(i_2)	$r_1 < 0 \vee$ $0 \leq r_1 < i_2 \vee$ $(r_1 = i_2 \wedge i_2 < \lvert s_3 \rvert \wedge s_3[i_2] = v_1)$	$r_1 < 0 \vee$ $0 \leq r_1 < i_2 \vee$ $(r_1 = i_2 \wedge i_2 < s_3$.size$() \wedge s_3$.get$(i_2) = v_1)$
s_1.remove_at(i_1)	s_2.add_at(i_2, v_2)	$(i_1 < i_2 \wedge s_3[i_2 - 1] = v_2) \vee$ $(i_1 = i_2 \wedge s_1[i_1] = v_2) \vee$ $(i_1 > i_2 \wedge s_3[i_1] = s_1[i_1])$	$(i_1 < i_2 \wedge s_3$.get$(i_2 - 1) = v_2) \vee$ $(i_1 = i_2 \wedge s_1$.get$(i_1) = v_2) \vee$ $(i_1 > i_2 \wedge s_3$.get$(i_1) = s_1$.get$(i_1))$
	$r_2 = s_2$.indexOf(v_2)	$(r_2 < 0 \wedge s_1[i_1] \neq v_2) \vee$ $0 \leq r_2 < i_1 \vee$ $(r_2 = i_1 \wedge s_1[i_1] = v_2 \wedge i_1 < \lvert s_3 \rvert)$	$(r_2 < 0 \wedge s_1$.get$(i_1) \neq v_2) \vee$ $0 \leq r_2 < i_1 \vee$ $(r_2 = i_1 \wedge s_1$.get$(i_1) = v_2 \wedge i_1 < s_3$.size$())$
	s_2.remove_at(i_2)	$(i_1 < i_2 \wedge s_3[i_2 - 1] = s_2[i_2]) \vee$ $i_1 = i_2 \vee$ $(\lvert s_3 \rvert + 1 > i_1 > i_2 \wedge s_1[i_1] = s_3[i_1 - 1])$	$(i_1 < i_2 \wedge s_3$.get$(i_2 - 1) = s_2$.get$(i_2)) \vee$ $i_1 = i_2 \vee$ $(s_3$.size$() + 1 > i_1 > i_2 \wedge s_1$.get$(i_1) = s_3$.get$(i_1 - 1))$

Table 7. After Commutativity Conditions on ArrayList

The commutativity conditions assume the operations operate on the same data structure (operations on different data structures trivially commute). s_1 denotes the data structure state before the first operation executes; s_2 denotes the state of the same data structure after the first operation executes but before the second operation executes. Each commutativity condition in the table corresponds to the execution order of the operation in the first column followed by the operation in the second column.

The before condition tables are symmetric (for a given pair of operations, the commutativity conditions are the same for both execution orders). The between condition tables may be asymmetric if the commutativity condition references either the return value from the first operation (which is not available in the other execution order) or depends on the intermediate data structure state (which may be different in the other execution order). Similarly, the after tables may be asymmetric if the commutativity condition depends on the intermediate or final data structure states.

In general, the commutativity conditions take the form of a disjunction of clauses. Dropping clauses produces conservative sound commutativity conditions that may be easier (for static analyses) or more efficient (for dynamic checkers) to work with. Such commutativity conditions are, of course, no longer complete. If the dropped clauses usually have no effect on the value of the commutativity condition, the gain in ease of reasoning or efficiency may be worth the the loss of completeness.

For completeness, some of the between conditions query the initial state (the state before the first operation executes). For the same reason, some of the after conditions query the initial and/or between states. In practice, there are two ways to dynamically check such commutativity conditions: 1) perform the query before the operation executes and record the result for the commutativity condition to check after the operation executes, or 2) drop the clause containing the query from the commutativity condition and use the resulting simpler, conservative, but not complete commutativity condition that does not reference the initial and/or between states.

One particularly useful special case is when the commutativity condition is `true` — i.e., the operations commute regardless of the data structure state. For example, `add` operations typically commute with other `add` operations, `contains` operations typically commute with other `contains` operations, and `remove` operations typically commute with other `remove` operations. Such commutativity conditions are particularly easy to reason about at compile time since the compiler does not need to reason about the parameter values or state to find commuting operations.

In general, our data structures implement `add` and `remove` operations that return values. The `add` operation typically returns the element that was previously present in the abstract set or map (or `null` if no such element existed), while the `remove` operation returns the removed element. We have verified commutativity conditions for two variants of these operations — one in which the client records the return value (typically by assigning the return

Data Structure	Verification Time
Accumulator	0.8s
AssociationList	1m 35s
HashSet	44s
HashTable	3m 20s
ListSet	40s
ArrayList[1]	12m 18s (3m 04s, 27s)

Table 8. Commutativity Testing Method Verification Times

Proof Language Command	Count
`note`	128
`assuming`	51
`pickWitness`	22
Total	201

Table 9. Additional Jahob Proof Language Commands for Remaining 57 ArrayList Commutativity Testing Methods

value to a variable) and another in which the client discards the return value. Tables 2 through 7 present the commutativity conditions only for the variants that discard the return value; the complete tables available in Appendix A present commutativity conditions for both variants. Because clients that record the return values observe more information about the data structure, the commutativity conditions for these variants can be more complex. For example, the between commutativity condition for the `r1a = sa.add(v1)`, `r2a = sa.add(v2)` pair from the ListSet and HashSet data structures is `(v1 = v2 | r1a)` (i.e., either `v1` and `v2` are different or `v1` was already in the set before the first operation executed), while the commutativity condition for the `s.add(v1)`, `s.add(v2)` pair is simply `true`.

5.2 Verification of Commutativity Conditions

For HashSet, ListSet, AssociationList, HashTable, and Accumulator, all of the automatically generated commutativity testing methods verify as generated. Table 8 presents the time required to verify all of these automatically generated methods. The verification times are all quite reasonable — less than four minutes for all data structures except ArrayList.

For ArrayList 429 of the 486 methods verify as generated. The entry for ArrayList in Table 8 indicates that Jahob spent 12m 18s attempting to verify all 486 automatically generated methods (with the majority of this time spent waiting for the Jahob integrated reasoning systems to time out as they try, but fail, to verify the 57 methods that require additional proof commands), 3m 04s verifying the 429 methods that verify as generated, and 28s verifying the remaining 57 methods after the addition of the required Jahob proof commands.

In general, the commutativity conditions for ArrayList are substantially more complicated than for other the data structures. We attribute this complexity in part to the use of integer indexing and in part to the presence of operations (such as `add_at` and `remove_at`) that shift the indexing relationships across large regions of the data structure.

The verification of the remaining 57 ArrayList commutativity testing methods required the addition of 128 `note` commands, 51 `assuming` commands, and 22 `pickWitness` commands (see Table 9).

In general, the `note` command allows the developer to specify an intermediate formula for Jahob to prove. Jahob can then use this formula in subsequent proofs. In this way, the developer can identify a lemma structure that helps Jahob find the proof.

[1] The Z3 and CVC3 decision procedures were each given a 20-second timeout.

The `assuming` command allows the developer to prove formulas of the form $A \implies B$ (by assuming A, then using A to prove B). We typically use the `assuming` command when Jahob is unable to prove a goal B in one of the cases A of the commutativity condition (or, when proving completeness, the negation of the commutativity condition). Providing a proof of $A \implies B$ enables Jahob to prove this case.

The `pickWitness` command allows the developer to start with an existentially quantified formula, name an element for which the formula holds, then remove the quantifier and use the resulting formula in a subsequent proof. We typically use this command when the commutativity condition (or its negation) contains an existential quantifier and we need to use the commutativity condition to prove a goal.

5.3 Proving the Remaining 57 Methods

The 57 remaining methods fall naturally into four categories. Each requires the proof language commands to manipulate either an existentially quantified formula or the negation of such a formula.

12 of the 57 methods are a soundness testing method for a combination of either `add_at(i,v1)` or `remove_at(i)` with either `indexOf(v2)` or `lastIndexOf(v2)`. The commutativity condition is either a between or after condition. We discuss the between condition for `add_at(i,v1)` with `indexOf(v2)`. The other combinations are similar. One of the cases of the commutativity condition states that `v2` is not present in the intermediate state of the map (so that `indexOf(v2)` returns -1). In this case Jahob must prove that the element is also not present in the initial state before `add_at(i,v1)` executes (so that the return value of `indexOf(v2)` is -1 in both execution orders). In effect, Jahob must prove that if the element is not present in the intermediate state, it is also not present in the initial state — in other words, Jahob must prove that the negation of one existentially quantified formula implies the negation of another existentially quantified formula. To enable Jahob to prove this fact, we use an `assuming` command, a `pickWitness` command, and several `note` commands to prove the contraposition (i.e., that if the element is present in the initial state, then it is also present in the intermediate state). A key step in the proof of the contraposition involves the identification of the new position of `v2` in the array after the `add_at(i,v1)` shifts it over (Jahob can automatically prove the cases when it is not shifted).

8 of the 57 methods are a soundness testing method for combinations of `remove_at(i)` with `indexOf(v)`. In these methods Jahob must prove that the return value of `indexOf(v)` is the same in both execution orders. The proof involves a case analysis of the initial state of the ArrayList. In one of the cases, the initial state contains two adjacent copies of `v`: one at location `i` and the other at location `i+1`. In this case, `remove_at(i)` removes the first occurrence of `v`, leaving the second occurrence of `v` in location `i`. In both execution orders `indexOf(v)` returns `i` (but `i` references conceptually different versions of `v` in the two execution orders). Jahob is unable to prove this fact without help. The addition of a `note` command that identifies the case and the new position of the second `v` after the `remove_at(v)` operation executes enables Jahob to complete the proof. The formula that identifies the case is the negation of a complex existentially quantified formula.

20 of the 57 methods are a completeness testing method for various combinations of `add_at(i,v)`, `remove_at(i,v)`, and `set(i,v)`. In these methods Jahob must prove that the two final abstract states are different. In general, Jahob accomplishes such a proof by finding an element that is present in one abstract state but not the other. In some cases, however, Jahob is unable to find such an element. The addition of an `assuming` command (which identifies the case) and `note` commands that identify the element and help Jahob prove which abstract state contains the element and

Operation		Inverse Operation
Accumulator	a.increase(v)	a.increase($-v$)
ListSet HashSet	$r = s$.add(v)	if $r = true$ then s.remove(v)
	$r = s$.remove(v)	if $r = true$ then s.add(v)
AssociationList HashTable	$r = d$.put(k, v)	if $r \neq null$ then d.put(k, r) else d.remove(k)
	$r = d$.remove(k)	if $r \neq null$ then d.put(k, r)
ArrayList	a.add_at(i, v)	a.remove_at(i)
	$r = a$.remove_at(i)	a.add_at(i, r)
	$r = a$.set(i, v)	a.set(i, r)

Table 10. Inverse Operations

which does not enables Jahob to complete the case analysis. The relevant formula identifying the case is existentially quantified.

17 of the 57 methods are a completeness testing method for combinations of either `add_at(i,v1)` or `remove_at(i)` with either `indexOf(v2)` or `lastIndexOf(v2)`. Recall that `add_at(i,v1)` shifts the region of the map above `i` up to make space for `v1` at index `i`. Similarly, `remove_at(i)` shifts the region of the map above `i` down to fill the hole left by the removed element. The verification of the completeness testing method involves a case analysis of the relative positions of the inserted or removed element and the element `v2` (whose index is returned by `indexOf(v2)` or `lastIndexOf(v2)`). In one of the cases Jahob is unable to reason successfully about these relative positions. The addition of an `assuming` command (which identifies the case) and a `note` command that identifies the precise position of `v2` (this `note` command follows from the formula which identifies the case) enables Jahob to complete the case analysis. Once again, the formula identifying the relevant case is existentially quantified.

5.4 Inverse Operations

Table 10 presents, for every operation that changes the data structure's abstract state, the corresponding inverse operation that rolls back the effect of the first operation to restore the original abstract state. Note that some of the inverse operations use the return value from the first operation. The inverse of the `put(k,v)` AssociationList and HashTable operation, for example, checks the return value to see if the `put(k,v)` operation replaced a previously mapped value for `k`. If so, it uses another `put` operation to restore the previously mapped value. If not, it simply removes the mapping from `k` to `v`. Any system that applies such inverse operations must therefore store the return value from the first operation so that it can provide the return value to the corresponding inverse operation. All of the eight inverse testing methods verified as generated without the need for additional Jahob proof commands.

6. Related Work

We have already surveyed related work in detecting commuting operations (Section 1.1). Commuting operations can also be used to simplify correctness proofs of parallel programs [6]. The basic idea is to use commutativity information to enable *reduction* — obtaining larger-grained atomic blocks by showing that finer-grain statements adjacent in one thread commute with statements in other threads. The specific method uses a form of computation abstraction (replacing statements with statements that have more behaviors) to enhance their ability to obtain statements that commute with other statements. While this may increase the possible behaviors of the program, the idea is to prove assertions at the end of the program. If these assertions are valid under the extended set of behaviors of the abstracted program, they are also valid for the original program.

The research presented in this paper uses a different form of abstraction (data abstraction as opposed to computation abstraction) for a different purpose (reasoning about the semantic equivalence of commuting and inverse operations on linked data structures). Our results may, however, enhance the effectiveness of techniques that reason about explicitly parallel programs — they provide such reasoning techniques with useful commutativity and inverse information about operations that manipulate linked data structures.

7. Conclusion

Commuting operations, commutativity conditions, and inverse operations play an important role in a broad range of current and envisioned static reasoning systems and parallel programs, languages, and systems. We have presented new techniques for verifying semantic commutativity conditions and inverse operations for linked data structures. Our results show that these techniques can effectively verify inverse operations and sound and complete commutativity conditions for a collection of challenging linked data structures. Our results therefore provide a useful foundation that others can build on as they develop static reasoning systems and parallel programs, languages, and systems.

References

[1] F. Aleen and N. Clark. Commutativity analysis for software parallelization: letting program transformations see the big picture. In *Proceeding of the 14th international conference on Architectural support for programming languages and operating systems*, ASPLOS '09, pages 241–252. ACM, 2009.

[2] R. L. Bocchino, Jr., V. S. Adve, D. Dig, S. V. Adve, S. Heumann, R. Komuravelli, J. Overbey, P. Simmons, H. Sung, and M. Vakilian. A type and effect system for deterministic parallel java. In *Proceeding of the 24th ACM SIGPLAN conference on Object oriented programming systems languages and applications*, OOPSLA '09, pages 97–116. ACM, 2009.

[3] C. Bouillaguet, V. Kuncak, T. Wies, K. Zee, and M. Rinard. Using first-order theorem provers in the jahob data structure verification system. In *Proceedings of the 8th international conference on Verification, model checking, and abstract interpretation*, VMCAI '07, pages 74–88. Springer-Verlag, 2007.

[4] M. J. Bridges, N. Vachharajani, Y. Zhang, T. Jablin, and D. I. August. Revisiting the sequential programming model for the multicore era. *IEEE Micro*, 28:12–20, January 2008.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009.

[6] T. Elmas, S. Qadeer, and S. Tasiran. A calculus of atomic actions. In *Proceedings of the 36th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, POPL '09, pages 2–15. ACM, 2009.

[7] A. Fekete, N. A. Lynch, M. Merritt, and W. E. Weihl. Commutativity-based locking for nested transactions. In *Proceedings of the Third International Workshop on Persistent Object Systems*, pages 319–340. Springer-Verlag, 1989.

[8] J. Gray and A. Reuter. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., 1st edition, 1992.

[9] S. Gulwani and G. C. Necula. Precise interprocedural analysis using random interpretation. In *Proceedings of the 32nd ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, POPL '05, pages 324–337. ACM, 2005.

[10] M. Kulkarni, K. Pingali, B. Walter, G. Ramanarayanan, K. Bala, and L. P. Chew. Optimistic parallelism requires abstractions. In *Proceedings of the 2007 ACM SIGPLAN conference on Programming language design and implementation*, PLDI '07, pages 211–222. ACM, 2007.

[11] M. Kulkarni, D. Prountzos, D. Nguyen, and K. Pingali. Defining and implementing commutativity conditions for parallel execution. Technical Report TR-ECE-09-11, School of Electrial and Computer Engineering, Purdue University, August 2009.

[12] M. C. Rinard and P. C. Diniz. Commutativity analysis: a new analysis technique for parallelizing compilers. *ACM Trans. Program. Lang. Syst.*, 19:942–991, November 1997.

[13] M. C. Rinard and M. S. Lam. The design, implementation, and evaluation of jade. *ACM Trans. Program. Lang. Syst.*, 20:483–545, May 1998.

[14] M. Sagiv, T. Reps, and R. Wilhelm. Parametric shape analysis via 3-valued logic. *ACM Trans. Program. Lang. Syst.*, 24:217–298, May 2002.

[15] N. Shavit and D. Touitou. Software transactional memory. In *Proceedings of the fourteenth annual ACM symposium on Principles of distributed computing*, PODC '95, pages 204–213. ACM, 1995.

[16] W. E. Weihl. Commutativity-based concurrency control for abstract data types. *IEEE Trans. Comput.*, 37:1488–1505, December 1988.

[17] K. Zee, V. Kuncak, and M. C. Rinard. Full functional verification of linked data structures. In *Proceedings of the 2008 ACM SIGPLAN conference on Programming language design and implementation*, PLDI '08, pages 349–361. ACM, 2008.

[18] K. Zee, V. Kuncak, and M. C. Rinard. An integrated proof language for imperative programs. In *Proceedings of the 2009 ACM SIGPLAN conference on Programming language design and implementation*, PLDI '09, pages 338–351. ACM, 2009.

A. Commutativity Conditions

A.1 Accumulator

Table 11: Commutativity Conditions on Accumulator

Methods		Commutativity Condition	
$s_1.\texttt{increase}(v_1)$	$s_2.\texttt{increase}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{read}()$	$v_1 = 0$	$v_1 = 0$
$r_1 = s_1.\texttt{read}()$	$s_2.\texttt{increase}(v_2)$	$v_2 = 0$	$v_2 = 0$
	$r_2 = s_2.\texttt{read}()$	$true$	$true$

A.2 ListSet and HashSet

A.2.1 Before Commutativity Conditions

Table 12: Before Commutativity Conditions on ListSet and HashSet

Methods		Commutativity Condition	
$r_1 = s_1.\texttt{add}(v_1)$	$r_2 = s_2.\texttt{add}(v_2)$	$v_1 \neq v_2 \vee v_1 \in s_1$	$v_1 \neq v_2 \vee s_1.\texttt{contains}(v_1) = true$
	$s_2.\texttt{add}(v_2)$	$v_1 \neq v_2 \vee v_1 \in s_1$	$v_1 \neq v_2 \vee s_1.\texttt{contains}(v_1) = true$
	$r_2 = s_2.\texttt{contains}(v_2)$	$v_1 \neq v_2 \vee v_1 \in s_1$	$v_1 \neq v_2 \vee s_1.\texttt{contains}(v_1) = true$
	$r_2 = s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2$	$v_1 \neq v_2$
	$s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2$	$v_1 \neq v_2$
	$r_2 = s_2.\texttt{size}()$	$v_1 \in s_1$	$s_1.\texttt{contains}(v_1) = true$
$s_1.\texttt{add}(v_1)$	$r_2 = s_2.\texttt{add}(v_2)$	$v_1 \neq v_2 \vee v_1 \in s_1$	$v_1 \neq v_2 \vee s_1.\texttt{contains}(v_1) = true$
	$s_2.\texttt{add}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{contains}(v_2)$	$v_1 \neq v_2 \vee v_1 \in s_1$	$v_1 \neq v_2 \vee s_1.\texttt{contains}(v_1) = true$
	$r_2 = s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2$	$v_1 \neq v_2$
	$s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2$	$v_1 \neq v_2$
	$r_2 = s_2.\texttt{size}()$	$v_1 \in s_1$	$s_1.\texttt{contains}(v_1) = true$
$r_1 = s_1.\texttt{contains}(v_1)$	$r_2 = s_2.\texttt{add}(v_2)$	$v_1 \neq v_2 \vee v_1 \in s_1$	$v_1 \neq v_2 \vee s_1.\texttt{contains}(v_1) = true$
	$s_2.\texttt{add}(v_2)$	$v_1 \neq v_2 \vee v_1 \in s_1$	$v_1 \neq v_2 \vee s_1.\texttt{contains}(v_1) = true$
	$r_2 = s_2.\texttt{contains}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2 \vee v_1 \notin s_1$	$v_1 \neq v_2 \vee s_1.\texttt{contains}(v_1) = false$
	$s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2 \vee v_1 \notin s_1$	$v_1 \neq v_2 \vee s_1.\texttt{contains}(v_1) = false$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$
$r_1 = s_1.\texttt{remove}(v_1)$	$r_2 = s_2.\texttt{add}(v_2)$	$v_1 \neq v_2$	$v_1 \neq v_2$
	$s_2.\texttt{add}(v_2)$	$v_1 \neq v_2$	$v_1 \neq v_2$
	$r_2 = s_2.\texttt{contains}(v_2)$	$v_1 \neq v_2 \vee v_1 \notin s_1$	$v_1 \neq v_2 \vee s_1.\texttt{contains}(v_1) = false$
	$r_2 = s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2 \vee v_1 \notin s_1$	$v_1 \neq v_2 \vee s_1.\texttt{contains}(v_1) = false$
	$s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2 \vee v_1 \notin s_1$	$v_1 \neq v_2 \vee s_1.\texttt{contains}(v_1) = false$
	$r_2 = s_2.\texttt{size}()$	$v_1 \notin s_1$	$s_1.\texttt{contains}(v_1) = false$
$s_1.\texttt{remove}(v_1)$	$r_2 = s_2.\texttt{add}(v_2)$	$v_1 \neq v_2$	$v_1 \neq v_2$
	$s_2.\texttt{add}(v_2)$	$v_1 \neq v_2$	$v_1 \neq v_2$
	$r_2 = s_2.\texttt{contains}(v_2)$	$v_1 \neq v_2 \vee v_1 \notin s_1$	$v_1 \neq v_2 \vee s_1.\texttt{contains}(v_1) = false$
	$r_2 = s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2 \vee v_1 \notin s_1$	$v_1 \neq v_2 \vee s_1.\texttt{contains}(v_1) = false$
	$s_2.\texttt{remove}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{size}()$	$v_1 \notin s_1$	$s_1.\texttt{contains}(v_1) = false$
$r_1 = s_1.\texttt{size}()$	$r_2 = s_2.\texttt{add}(v_2)$	$v_2 \in s_1$	$s_1.\texttt{contains}(v_2) = true$
	$s_2.\texttt{add}(v_2)$	$v_2 \in s_1$	$s_1.\texttt{contains}(v_2) = true$
	$r_2 = s_2.\texttt{contains}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{remove}(v_2)$	$v_2 \notin s_1$	$s_1.\texttt{contains}(v_2) = false$
	$s_2.\texttt{remove}(v_2)$	$v_2 \notin s_1$	$s_1.\texttt{contains}(v_2) = false$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$

A.2.2 Between Commutativity Conditions

Table 13: Between Commutativity Conditions on ListSet and HashSet

Methods		Commutativity Condition	
$r_1 = s_1.\texttt{add}(v_1)$	$r_2 = s_2.\texttt{add}(v_2)$	$v_1 \neq v_2 \vee r_1 = false$	$v_1 \neq v_2 \vee r_1 = false$
	$s_2.\texttt{add}(v_2)$	$v_1 \neq v_2 \vee r_1 = false$	$v_1 \neq v_2 \vee r_1 = false$

Table 13: Between Commutativity Conditions on ListSet and HashSet (Cont.)

	$r_2 = s_2.\texttt{contains}(v_2)$	$v_1 \neq v_2 \vee r_1 = false$	$v_1 \neq v_2 \vee r_1 = false$
	$r_2 = s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2$	$v_1 \neq v_2$
	$s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2$	$v_1 \neq v_2$
	$r_2 = s_2.\texttt{size}()$	$r_1 = false$	$r_1 = false$
$s_1.\texttt{add}(v_1)$	$r_2 = s_2.\texttt{add}(v_2)$	$v_1 \neq v_2 \vee v_1 \in s_1$	$v_1 \neq v_2 \vee s_1.\texttt{contains}(v_1) = true$
	$s_2.\texttt{add}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{contains}(v_2)$	$v_1 \neq v_2 \vee v_1 \in s_1$	$v_1 \neq v_2 \vee s_1.\texttt{contains}(v_1) = true$
	$r_2 = s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2$	$v_1 \neq v_2$
	$s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2$	$v_1 \neq v_2$
	$r_2 = s_2.\texttt{size}()$	$v_1 \in s_1$	$s_1.\texttt{contains}(v_1) = true$
$r_1 = s_1.\texttt{contains}(v_1)$	$r_2 = s_2.\texttt{add}(v_2)$	$v_1 \neq v_2 \vee r_1 = true$	$v_1 \neq v_2 \vee r_1 = true$
	$s_2.\texttt{add}(v_2)$	$v_1 \neq v_2 \vee r_1 = true$	$v_1 \neq v_2 \vee r_1 = true$
	$r_2 = s_2.\texttt{contains}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2 \vee r_1 = false$	$v_1 \neq v_2 \vee r_1 = false$
	$s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2 \vee r_1 = false$	$v_1 \neq v_2 \vee r_1 = false$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$
$r_1 = s_1.\texttt{remove}(v_1)$	$r_2 = s_2.\texttt{add}(v_2)$	$v_1 \neq v_2$	$v_1 \neq v_2$
	$s_2.\texttt{add}(v_2)$	$v_1 \neq v_2$	$v_1 \neq v_2$
	$r_2 = s_2.\texttt{contains}(v_2)$	$v_1 \neq v_2 \vee r_1 = false$	$v_1 \neq v_2 \vee r_1 = false$
	$r_2 = s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2 \vee r_1 = false$	$v_1 \neq v_2 \vee r_1 = false$
	$s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2 \vee r_1 = false$	$v_1 \neq v_2 \vee r_1 = false$
	$r_2 = s_2.\texttt{size}()$	$r_1 = false$	$r_1 = false$
$s_1.\texttt{remove}(v_1)$	$r_2 = s_2.\texttt{add}(v_2)$	$v_1 \neq v_2$	$v_1 \neq v_2$
	$s_2.\texttt{add}(v_2)$	$v_1 \neq v_2$	$v_1 \neq v_2$
	$r_2 = s_2.\texttt{contains}(v_2)$	$v_1 \neq v_2 \vee v_1 \notin s_1$	$v_1 \neq v_2 \vee s_1.\texttt{contains}(v_1) = false$
	$r_2 = s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2 \vee v_1 \notin s_1$	$v_1 \neq v_2 \vee s_1.\texttt{contains}(v_1) = false$
	$s_2.\texttt{remove}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{size}()$	$v_1 \notin s_1$	$s_1.\texttt{contains}(v_1) = false$
$r_1 = s_1.\texttt{size}()$	$r_2 = s_2.\texttt{add}(v_2)$	$v_2 \in s_2$	$s_2.\texttt{contains}(v_2) = true$
	$s_2.\texttt{add}(v_2)$	$v_2 \in s_2$	$s_2.\texttt{contains}(v_2) = true$
	$r_2 = s_2.\texttt{contains}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{remove}(v_2)$	$v_2 \notin s_2$	$s_2.\texttt{contains}(v_2) = false$
	$s_2.\texttt{remove}(v_2)$	$v_2 \notin s_2$	$s_2.\texttt{contains}(v_2) = false$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$

A.2.3 After Commutativity Conditions

Table 14: After Commutativity Conditions on ListSet and HashSet

Methods		Commutativity Condition	
$r_1 = s_1.\texttt{add}(v_1)$	$r_2 = s_2.\texttt{add}(v_2)$	$v_1 \neq v_2 \vee r_1 = false$	$v_1 \neq v_2 \vee r_1 = false$
	$s_2.\texttt{add}(v_2)$	$v_1 \neq v_2 \vee r_1 = false$	$v_1 \neq v_2 \vee r_1 = false$
	$r_2 = s_2.\texttt{contains}(v_2)$	$v_1 \neq v_2 \vee r_1 = false$	$v_1 \neq v_2 \vee r_1 = false$
	$r_2 = s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2$	$v_1 \neq v_2$
	$s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2$	$v_1 \neq v_2$
	$r_2 = s_2.\texttt{size}()$	$r_1 = false$	$r_1 = false$
$s_1.\texttt{add}(v_1)$	$r_2 = s_2.\texttt{add}(v_2)$	$v_1 \neq v_2 \vee v_1 \in s_1$	$v_1 \neq v_2 \vee s_1.\texttt{contains}(v_1) = true$
	$s_2.\texttt{add}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{contains}(v_2)$	$v_1 \neq v_2 \vee v_1 \in s_1$	$v_1 \neq v_2 \vee s_1.\texttt{contains}(v_1) = true$
	$r_2 = s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2$	$v_1 \neq v_2$
	$s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2$	$v_1 \neq v_2$
	$r_2 = s_2.\texttt{size}()$	$v_1 \in s_1$	$s_1.\texttt{contains}(v_1) = true$
$r_1 = s_1.\texttt{contains}(v_1)$	$r_2 = s_2.\texttt{add}(v_2)$	$v_1 \neq v_2 \vee r_1 = true$	$v_1 \neq v_2 \vee r_1 = true$
	$s_2.\texttt{add}(v_2)$	$v_1 \neq v_2 \vee r_1 = true$	$v_1 \neq v_2 \vee r_1 = true$
	$r_2 = s_2.\texttt{contains}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2 \vee r_1 = false$	$v_1 \neq v_2 \vee r_1 = false$
	$s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2 \vee r_1 = false$	$v_1 \neq v_2 \vee r_1 = false$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$
$r_1 = s_1.\texttt{remove}(v_1)$	$r_2 = s_2.\texttt{add}(v_2)$	$v_1 \neq v_2$	$v_1 \neq v_2$
	$s_2.\texttt{add}(v_2)$	$v_1 \neq v_2$	$v_1 \neq v_2$
	$r_2 = s_2.\texttt{contains}(v_2)$	$v_1 \neq v_2 \vee r_1 = false$	$v_1 \neq v_2 \vee r_1 = false$
	$r_2 = s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2 \vee r_1 = false$	$v_1 \neq v_2 \vee r_1 = false$

Table 14: After Commutativity Conditions on ListSet and HashSet (Cont.)

	$s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2 \vee r_1 = false$	$v_1 \neq v_2 \vee r_1 = false$
	$r_2 = s_2.\texttt{size}()$	$r_1 = false$	$r_1 = false$
$s_1.\texttt{remove}(v_1)$	$r_2 = s_2.\texttt{add}(v_2)$	$v_1 \neq v_2$	$v_1 \neq v_2$
	$s_2.\texttt{add}(v_2)$	$v_1 \neq v_2$	$v_1 \neq v_2$
	$r_2 = s_2.\texttt{contains}(v_2)$	$v_1 \neq v_2 \vee v_1 \notin s_1$	$v_1 \neq v_2 \vee s_1.\texttt{contains}(v_1) = false$
	$r_2 = s_2.\texttt{remove}(v_2)$	$v_1 \neq v_2 \vee v_1 \notin s_1$	$v_1 \neq v_2 \vee s_1.\texttt{contains}(v_1) = false$
	$s_2.\texttt{remove}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{size}()$	$v_1 \notin s_1$	$s_1.\texttt{contains}(v_1) = false$
$r_1 = s_1.\texttt{size}()$	$r_2 = s_2.\texttt{add}(v_2)$	$r_2 = false$	$r_2 = false$
	$s_2.\texttt{add}(v_2)$	$v_2 \in s_2$	$s_2.\texttt{contains}(v_2) = true$
	$r_2 = s_2.\texttt{contains}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{remove}(v_2)$	$r_2 = false$	$r_2 = false$
	$s_2.\texttt{remove}(v_2)$	$v_2 \notin s_2$	$s_2.\texttt{contains}(v_2) = false$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$

A.3 AssociationList and HashTable

A.3.1 Before Commutativity Conditions

Table 15: Before Commutativity Conditions on AssociationList and HashTable

Methods		Commutativity Condition	
$r_1 = s_1.\texttt{containsKey}(k_1)$	$r_2 = s_2.\texttt{containsKey}(k_2)$	$true$	$true$
	$r_2 = s_2.\texttt{get}(k_2)$	$true$	$true$
	$r_2 = s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2 \vee \langle k_1, \_\rangle \in s_1$	$k_1 \neq k_2 \vee s_1.\texttt{containsKey}(k_1) = true$
	$s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2 \vee \langle k_1, \_\rangle \in s_1$	$k_1 \neq k_2 \vee s_1.\texttt{containsKey}(k_1) = true$
	$r_2 = s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, \_\rangle \notin s_1$	$k_1 \neq k_2 \vee s_1.\texttt{containsKey}(k_1) = false$
	$s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, \_\rangle \notin s_1$	$k_1 \neq k_2 \vee s_1.\texttt{containsKey}(k_1) = false$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$
$r_1 = s_1.\texttt{get}(k_1)$	$r_2 = s_2.\texttt{containsKey}(k_2)$	$true$	$true$
	$r_2 = s_2.\texttt{get}(k_2)$	$true$	$true$
	$r_2 = s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2 \vee \langle k_1, v_2\rangle \in s_1$	$k_1 \neq k_2 \vee s_1.\texttt{get}(k_1) = v_2$
	$s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2 \vee \langle k_1, v_2\rangle \in s_1$	$k_1 \neq k_2 \vee s_1.\texttt{get}(k_1) = v_2$
	$r_2 = s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, \_\rangle \notin s_1$	$k_1 \neq k_2 \vee s_1.\texttt{containsKey}(k_1) = false$
	$s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, \_\rangle \notin s_1$	$k_1 \neq k_2 \vee s_1.\texttt{containsKey}(k_1) = false$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$
$r_1 = s_1.\texttt{put}(k_1, v_1)$	$r_2 = s_2.\texttt{containsKey}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, \_\rangle \in s_1$	$k_1 \neq k_2 \vee s_1.\texttt{containsKey}(k_1) = true$
	$r_2 = s_2.\texttt{get}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, v_1\rangle \in s_1$	$k_1 \neq k_2 \vee s_1.\texttt{get}(k_1) = v_1$
	$r_2 = s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2 \vee (\langle k_1, v_1\rangle \in s_1 \wedge v_1 = v_2)$	$k_1 \neq k_2 \vee (s_1.\texttt{get}(k_1) = v_1 \wedge v_1 = v_2)$
	$s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2 \vee (\langle k_1, v_1\rangle \in s_1 \wedge v_1 = v_2)$	$k_1 \neq k_2 \vee (s_1.\texttt{get}(k_1) = v_1 \wedge v_1 = v_2)$
	$r_2 = s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2$	$k_1 \neq k_2$
	$s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2$	$k_1 \neq k_2$
	$r_2 = s_2.\texttt{size}()$	$\langle k_1, \_\rangle \in s_1$	$s_1.\texttt{containsKey}(k_1) = true$
$s_1.\texttt{put}(k_1, v_1)$	$r_2 = s_2.\texttt{containsKey}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, \_\rangle \in s_1$	$k_1 \neq k_2 \vee s_1.\texttt{containsKey}(k_1) = true$
	$r_2 = s_2.\texttt{get}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, v_1\rangle \in s_1$	$k_1 \neq k_2 \vee s_1.\texttt{get}(k_1) = v_1$
	$r_2 = s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2 \vee (\langle k_1, v_1\rangle \in s_1 \wedge v_1 = v_2)$	$k_1 \neq k_2 \vee (s_1.\texttt{get}(k_1) = v_1 \wedge v_1 = v_2)$
	$s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2 \vee v_1 = v_2$	$k_1 \neq k_2 \vee v_1 = v_2$
	$r_2 = s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2$	$k_1 \neq k_2$
	$s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2$	$k_1 \neq k_2$
	$r_2 = s_2.\texttt{size}()$	$\langle k_1, \_\rangle \in s_1$	$s_1.\texttt{containsKey}(k_1) = true$
$r_1 = s_1.\texttt{remove}(k_1)$	$r_2 = s_2.\texttt{containsKey}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, \_\rangle \notin s_1$	$k_1 \neq k_2 \vee s_1.\texttt{containsKey}(k_1) = false$
	$r_2 = s_2.\texttt{get}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, \_\rangle \notin s_1$	$k_1 \neq k_2 \vee s_1.\texttt{containsKey}(k_1) = false$
	$r_2 = s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2$	$k_1 \neq k_2$
	$s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2$	$k_1 \neq k_2$
	$r_2 = s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, \_\rangle \notin s_1$	$k_1 \neq k_2 \vee s_1.\texttt{containsKey}(k_1) = false$
	$s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, \_\rangle \notin s_1$	$k_1 \neq k_2 \vee s_1.\texttt{containsKey}(k_1) = false$
	$r_2 = s_2.\texttt{size}()$	$\langle k_1, \_\rangle \notin s_1$	$s_1.\texttt{containsKey}(k_1) = false$
$s_1.\texttt{remove}(k_1)$	$r_2 = s_2.\texttt{containsKey}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, \_\rangle \notin s_1$	$k_1 \neq k_2 \vee s_1.\texttt{containsKey}(k_1) = false$
	$r_2 = s_2.\texttt{get}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, \_\rangle \notin s_1$	$k_1 \neq k_2 \vee s_1.\texttt{containsKey}(k_1) = false$
	$r_2 = s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2$	$k_1 \neq k_2$
	$s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2$	$k_1 \neq k_2$
	$r_2 = s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, \_\rangle \notin s_1$	$k_1 \neq k_2 \vee s_1.\texttt{containsKey}(k_1) = false$
	$s_2.\texttt{remove}(k_2)$	$true$	$true$

Table 15: Before Commutativity Conditions on AssociationList and HashTable (Cont.)

	$r_2 = s_2.\texttt{size}()$	$\langle k_1, \_\rangle \notin s_1$	$s_1.\texttt{containsKey}(k_1) = false$
$r_1 = s_1.\texttt{size}()$	$r_2 = s_2.\texttt{containsKey}(k_2)$	$true$	$true$
	$r_2 = s_2.\texttt{get}(k_2)$	$true$	$true$
	$r_2 = s_2.\texttt{put}(k_2, v_2)$	$\langle k_2, \_\rangle \in s_1$	$s_1.\texttt{containsKey}(k_2) = true$
	$s_2.\texttt{put}(k_2, v_2)$	$\langle k_2, \_\rangle \in s_1$	$s_1.\texttt{containsKey}(k_2) = true$
	$r_2 = s_2.\texttt{remove}(k_2)$	$\langle k_2, \_\rangle \notin s_1$	$s_1.\texttt{containsKey}(k_2) = false$
	$s_2.\texttt{remove}(k_2)$	$\langle k_2, \_\rangle \notin s_1$	$s_1.\texttt{containsKey}(k_2) = false$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$

A.3.2 Between Commutativity Conditions

Table 16: Between Commutativity Conditions on AssociationList and HashTable

Methods		Commutativity Condition	
$r_1 = s_1.\texttt{containsKey}(k_1)$	$r_2 = s_2.\texttt{containsKey}(k_2)$	$true$	$true$
	$r_2 = s_2.\texttt{get}(k_2)$	$true$	$true$
	$r_2 = s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2 \vee r_1 = true$	$k_1 \neq k_2 \vee r_1 = true$
	$s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2 \vee r_1 = true$	$k_1 \neq k_2 \vee r_1 = true$
	$r_2 = s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2 \vee r_1 = false$	$k_1 \neq k_2 \vee r_1 = false$
	$s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2 \vee r_1 = false$	$k_1 \neq k_2 \vee r_1 = false$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$
$r_1 = s_1.\texttt{get}(k_1)$	$r_2 = s_2.\texttt{containsKey}(k_2)$	$true$	$true$
	$r_2 = s_2.\texttt{get}(k_2)$	$true$	$true$
	$r_2 = s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2 \vee r_1 = v_2$	$k_1 \neq k_2 \vee r_1 = v_2$
	$s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2 \vee r_1 = v_2$	$k_1 \neq k_2 \vee r_1 = v_2$
	$r_2 = s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2 \vee r_1 = null$	$k_1 \neq k_2 \vee r_1 = null$
	$s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2 \vee r_1 = null$	$k_1 \neq k_2 \vee r_1 = null$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$
$r_1 = s_1.\texttt{put}(k_1, v_1)$	$r_2 = s_2.\texttt{containsKey}(k_2)$	$k_1 \neq k_2 \vee r_1 \neq null$	$k_1 \neq k_2 \vee r_1 \neq null$
	$r_2 = s_2.\texttt{get}(k_2)$	$k_1 \neq k_2 \vee r_1 = v_1$	$k_1 \neq k_2 \vee r_1 = v_1$
	$r_2 = s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2 \vee (r_1 = v_1 \wedge v_1 = v_2)$	$k_1 \neq k_2 \vee (r_1 = v_1 \wedge v_1 = v_2)$
	$s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2 \vee (r_1 = v_1 \wedge v_1 = v_2)$	$k_1 \neq k_2 \vee (r_1 = v_1 \wedge v_1 = v_2)$
	$r_2 = s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2$	$k_1 \neq k_2$
	$s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2$	$k_1 \neq k_2$
	$r_2 = s_2.\texttt{size}()$	$r_1 \neq null$	$r_1 \neq null$
$s_1.\texttt{put}(k_1, v_1)$	$r_2 = s_2.\texttt{containsKey}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, \_\rangle \in s_1$	$k_1 \neq k_2 \vee s_1.\texttt{containsKey}(k_1) = true$
	$r_2 = s_2.\texttt{get}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, v_1\rangle \in s_1$	$k_1 \neq k_2 \vee s_1.\texttt{get}(k_1) = v_1$
	$r_2 = s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2 \vee (\langle k_1, v_1\rangle \in s_1 \wedge v_1 = v_2)$	$k_1 \neq k_2 \vee (s_1.\texttt{get}(k_1) = v_1 \wedge v_1 = v_2)$
	$s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2 \vee v_1 = v_2$	$k_1 \neq k_2 \vee v_1 = v_2$
	$r_2 = s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2$	$k_1 \neq k_2$
	$s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2$	$k_1 \neq k_2$
	$r_2 = s_2.\texttt{size}()$	$r_1 \neq null$	$r_1 \neq null$
$r_1 = s_1.\texttt{remove}(k_1)$	$r_2 = s_2.\texttt{containsKey}(k_2)$	$k_1 \neq k_2 \vee r_1 = null$	$k_1 \neq k_2 \vee r_1 = null$
	$r_2 = s_2.\texttt{get}(k_2)$	$k_1 \neq k_2 \vee r_1 = null$	$k_1 \neq k_2 \vee r_1 = null$
	$r_2 = s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2$	$k_1 \neq k_2$
	$s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2$	$k_1 \neq k_2$
	$r_2 = s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2 \vee r_1 = null$	$k_1 \neq k_2 \vee r_1 = null$
	$s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2 \vee r_1 = null$	$k_1 \neq k_2 \vee r_1 = null$
	$r_2 = s_2.\texttt{size}()$	$r_1 = null$	$r_1 = null$
$s_1.\texttt{remove}(k_1)$	$r_2 = s_2.\texttt{containsKey}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, \_\rangle \notin s_1$	$k_1 \neq k_2 \vee s_1.\texttt{containsKey}(k_1) = false$
	$r_2 = s_2.\texttt{get}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, \_\rangle \notin s_1$	$k_1 \neq k_2 \vee s_1.\texttt{containsKey}(k_1) = false$
	$r_2 = s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2$	$k_1 \neq k_2$
	$s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2$	$k_1 \neq k_2$
	$r_2 = s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, \_\rangle \notin s_1$	$k_1 \neq k_2 \vee s_1.\texttt{containsKey}(k_1) = false$
	$s_2.\texttt{remove}(k_2)$	$true$	$true$
	$r_2 = s_2.\texttt{size}()$	$\langle k_1, \_\rangle \notin s_1$	$s_1.\texttt{containsKey}(k_1) = false$
$r_1 = s_1.\texttt{size}()$	$r_2 = s_2.\texttt{containsKey}(k_2)$	$true$	$true$
	$r_2 = s_2.\texttt{get}(k_2)$	$true$	$true$
	$r_2 = s_2.\texttt{put}(k_2, v_2)$	$\langle k_2, \_\rangle \in s_2$	$s_2.\texttt{containsKey}(k_2) = true$
	$s_2.\texttt{put}(k_2, v_2)$	$\langle k_2, \_\rangle \in s_2$	$s_2.\texttt{containsKey}(k_2) = true$
	$r_2 = s_2.\texttt{remove}(k_2)$	$\langle k_2, \_\rangle \notin s_2$	$s_2.\texttt{containsKey}(k_2) = false$
	$s_2.\texttt{remove}(k_2)$	$\langle k_2, \_\rangle \notin s_2$	$s_2.\texttt{containsKey}(k_2) = false$

Table 16: Between Commutativity Conditions on AssociationList and HashTable (Cont.)

	$r_2 = s_2.\texttt{size}()$	$true$	$true$

A.3.3 After Commutativity Conditions

Table 17: After Commutativity Conditions on AssociationList and HashTable

Methods		Commutativity Condition	
$r_1 = s_1.\texttt{containsKey}(k_1)$	$r_2 = s_2.\texttt{containsKey}(k_2)$	$true$	$true$
	$r_2 = s_2.\texttt{get}(k_2)$	$true$	$true$
	$r_2 = s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2 \vee r_1 = true$	$k_1 \neq k_2 \vee r_1 = true$
	$s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2 \vee r_1 = true$	$k_1 \neq k_2 \vee r_1 = true$
	$r_2 = s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2 \vee r_1 = false$	$k_1 \neq k_2 \vee r_1 = false$
	$s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2 \vee r_1 = false$	$k_1 \neq k_2 \vee r_1 = false$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$
$r_1 = s_1.\texttt{get}(k_1)$	$r_2 = s_2.\texttt{containsKey}(k_2)$	$true$	$true$
	$r_2 = s_2.\texttt{get}(k_2)$	$true$	$true$
	$r_2 = s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2 \vee r_1 = v_2$	$k_1 \neq k_2 \vee r_1 = v_2$
	$s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2 \vee r_1 = v_2$	$k_1 \neq k_2 \vee r_1 = v_2$
	$r_2 = s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2 \vee r_1 = null$	$k_1 \neq k_2 \vee r_1 = null$
	$s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2 \vee r_1 = null$	$k_1 \neq k_2 \vee r_1 = null$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$
$r_1 = s_1.\texttt{put}(k_1, v_1)$	$r_2 = s_2.\texttt{containsKey}(k_2)$	$k_1 \neq k_2 \vee r_1 \neq null$	$k_1 \neq k_2 \vee r_1 \neq null$
	$r_2 = s_2.\texttt{get}(k_2)$	$k_1 \neq k_2 \vee r_1 = v_1$	$k_1 \neq k_2 \vee r_1 = v_1$
	$r_2 = s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2 \vee (r_1 = v_1 \wedge v_1 = v_2)$	$k_1 \neq k_2 \vee (r_1 = v_1 \wedge v_1 = v_2)$
	$s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2 \vee (r_1 = v_1 \wedge v_1 = v_2)$	$k_1 \neq k_2 \vee (r_1 = v_1 \wedge v_1 = v_2)$
	$r_2 = s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2$	$k_1 \neq k_2$
	$s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2$	$k_1 \neq k_2$
	$r_2 = s_2.\texttt{size}()$	$r_1 \neq null$	$r_1 \neq null$
$s_1.\texttt{put}(k_1, v_1)$	$r_2 = s_2.\texttt{containsKey}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, \_ \rangle \in s_1$	$k_1 \neq k_2 \vee s_1.\texttt{containsKey}(k_1) = true$
	$r_2 = s_2.\texttt{get}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, v_1 \rangle \in s_1$	$k_1 \neq k_2 \vee s_1.\texttt{get}(k_1) = v_1$
	$r_2 = s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2 \vee (\langle k_1, v_1 \rangle \in s_1 \wedge v_1 = v_2)$	$k_1 \neq k_2 \vee (s_1.\texttt{get}(k_1) = v_1 \wedge v_1 = v_2)$
	$s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2 \vee v_1 = v_2$	$k_1 \neq k_2 \vee v_1 = v_2$
	$r_2 = s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2$	$k_1 \neq k_2$
	$s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2$	$k_1 \neq k_2$
	$r_2 = s_2.\texttt{size}()$	$\langle k_1, \_ \rangle \in s_1$	$s_1.\texttt{containsKey}(k_1) = true$
$r_1 = s_1.\texttt{remove}(k_1)$	$r_2 = s_2.\texttt{containsKey}(k_2)$	$k_1 \neq k_2 \vee r_1 = null$	$k_1 \neq k_2 \vee r_1 = null$
	$r_2 = s_2.\texttt{get}(k_2)$	$k_1 \neq k_2 \vee r_1 = null$	$k_1 \neq k_2 \vee r_1 = null$
	$r_2 = s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2$	$k_1 \neq k_2$
	$s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2$	$k_1 \neq k_2$
	$r_2 = s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2 \vee r_1 = null$	$k_1 \neq k_2 \vee r_1 = null$
	$s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2 \vee r_1 = null$	$k_1 \neq k_2 \vee r_1 = null$
	$r_2 = s_2.\texttt{size}()$	$r_1 = null$	$r_1 = null$
$s_1.\texttt{remove}(k_1)$	$r_2 = s_2.\texttt{containsKey}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, \_ \rangle \notin s_1$	$k_1 \neq k_2 \vee s_1.\texttt{containsKey}(k_1) = false$
	$r_2 = s_2.\texttt{get}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, \_ \rangle \notin s_1$	$k_1 \neq k_2 \vee s_1.\texttt{containsKey}(k_1) = false$
	$r_2 = s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2$	$k_1 \neq k_2$
	$s_2.\texttt{put}(k_2, v_2)$	$k_1 \neq k_2$	$k_1 \neq k_2$
	$r_2 = s_2.\texttt{remove}(k_2)$	$k_1 \neq k_2 \vee \langle k_1, \_ \rangle \notin s_1$	$k_1 \neq k_2 \vee s_1.\texttt{containsKey}(k_1) = false$
	$s_2.\texttt{remove}(k_2)$	$true$	$true$
	$r_2 = s_2.\texttt{size}()$	$\langle k_1, \_ \rangle \notin s_1$	$s_1.\texttt{containsKey}(k_1) = false$
$r_1 = s_1.\texttt{size}()$	$r_2 = s_2.\texttt{containsKey}(k_2)$	$true$	$true$
	$r_2 = s_2.\texttt{get}(k_2)$	$true$	$true$
	$r_2 = s_2.\texttt{put}(k_2, v_2)$	$r_2 \neq null$	$r_2 \neq null$
	$s_2.\texttt{put}(k_2, v_2)$	$\langle k_2, \_ \rangle \in s_2$	$s_2.\texttt{containsKey}(k_2) = true$
	$r_2 = s_2.\texttt{remove}(k_2)$	$r_2 = null$	$r_2 = null$
	$s_2.\texttt{remove}(k_2)$	$\langle k_2, \_ \rangle \notin s_2$	$s_2.\texttt{containsKey}(k_2) = false$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$

A.4 ArrayList

A.4.1 Before Commutativity Conditions

Table 18: Before Commutativity Conditions on ArrayList

Methods		Commutativity Condition	
$s_1.\texttt{add\_at}(i_1, v_1)$	$s_2.\texttt{add\_at}(i_2, v_2)$	$(i_1 < i_2 \leq \lvert s_1\rvert \wedge s_1[i_2 - 1] = v_2) \vee$ $(i_1 = i_2 \wedge v_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_1[i_1 - 1] = v_1)$	$(i_1 < i_2 \leq s_1.\texttt{size}() \wedge s_1.\texttt{get}(i_2 - 1) = v_2) \vee$ $(i_1 = i_2 \wedge v_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_1.\texttt{get}(i_1 - 1) = v_1)$
	$r_2 = s_2.\texttt{get}(i_2)$	$(i_1 < i_2 < \lvert s_1\rvert \wedge s_1[i_2 - 1] = s_1[i_2]) \vee$ $(i_1 = i_2 < \lvert s_1\rvert \wedge s_1[i_1] = v_1) \vee$ $i_1 > i_2$	$(i_1 < i_2 < s_1.\texttt{size}() \wedge s_1.\texttt{get}(i_2 - 1) = s_1.\texttt{get}(i_2)) \vee$ $(i_1 = i_2 < s_1.\texttt{size}() \wedge s_1.\texttt{get}(i_1) = v_1) \vee$ $i_1 > i_2$
	$r_2 = s_2.\texttt{indexOf}(v_2)$	$(\neg(\exists i : s_1[i] = v_2) \wedge v_1 \neq v_2) \vee$ $(\exists i < i_1 : s_1[i] = v_2) \vee$ $(\neg(\exists i < i_1 : s_1[i] = v_2) \wedge s_1[i_1] = v_2 \wedge v_1 = v_2)$	$(s_1.\texttt{indexOf}(v_2) < 0 \wedge v_1 \neq v_2) \vee$ $0 \leq s_1.\texttt{indexOf}(v_2) < i_1 \vee$ $(s_1.\texttt{indexOf}(v_2) = i_1 \wedge v_1 = v_2)$
	$r_2 = s_2.\texttt{lastIndexOf}(v_2)$	$(\neg(\exists i : s_1[i] = v_2) \wedge v_1 \neq v_2) \vee$ $((\exists i < i_1 : s_1[i] = v_2) \wedge \neg(\exists i \geq i_1 : s_1[i] = v_2) \wedge v_1 \neq v_2)$	$(s_1.\texttt{lastIndexOf}(v_2) < 0 \wedge v_1 \neq v_2) \vee$ $(0 \leq s_1.\texttt{lastIndexOf}(v_2) < i_1 \wedge v_1 \neq v_2)$
	$r_2 = s_2.\texttt{remove\_at}(i_2)$	$(i_1 < i_2 < \lvert s_1\rvert \wedge s_1[i_2 - 1] = s_1[i_2]) \vee$ $(\lvert s_1\rvert - 1 \geq i_1 = i_2 \wedge s_1[i_1] = v_1) \vee$ $(\lvert s_1\rvert - 1 \geq i_1 > i_2 \wedge s_1[i_1] = v_1)$	$(i_1 < i_2 < s_1.\texttt{size}() \wedge s_1.\texttt{get}(i_2 - 1) = s_1.\texttt{get}(i_2)) \vee$ $(s_1.\texttt{size}() - 1 \geq i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = v_1) \vee$ $(s_1.\texttt{size}() - 1 \geq i_1 > i_2 \wedge s_1.\texttt{get}(i_1) = v_1)$
	$s_2.\texttt{remove\_at}(i_2)$	$(i_1 < i_2 < \lvert s_1\rvert \wedge s_1[i_2 - 1] = s_1[i_2]) \vee$ $(\lvert s_1\rvert - 1 \geq i_1 = i_2 \wedge s_1[i_1] = v_1) \vee$ $(\lvert s_1\rvert - 1 \geq i_1 > i_2 \wedge s_1[i_1] = v_1)$	$(i_1 < i_2 < s_1.\texttt{size}() \wedge s_1.\texttt{get}(i_2 - 1) = s_1.\texttt{get}(i_2)) \vee$ $(s_1.\texttt{size}() - 1 \geq i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = v_1) \vee$ $(s_1.\texttt{size}() - 1 \geq i_1 > i_2 \wedge s_1.\texttt{get}(i_1) = v_1)$
	$r_2 = s_2.\texttt{set}(i_2, v_2)$	$(i_1 < i_2 < \lvert s_1\rvert \wedge s_1[i_2 - 1] = s_1[i_2] = v_2) \vee$ $(i_1 = i_2 < \lvert s_1\rvert \wedge s_1[i_2] = v_1 = v_2) \vee$ $i_1 > i_2$	$(i_1 < i_2 < s_1.\texttt{size}() \wedge s_1.\texttt{get}(i_2 - 1) = s_1.\texttt{get}(i_2) = v_2) \vee$ $(i_1 = i_2 < s_1.\texttt{size}() \wedge s_1.\texttt{get}(i_2) = v_1 = v_2) \vee$ $i_1 > i_2$
	$s_2.\texttt{set}(i_2, v_2)$	$(i_1 < i_2 < \lvert s_1\rvert \wedge s_1[i_2 - 1] = s_1[i_2] = v_2) \vee$ $(i_1 = i_2 < \lvert s_1\rvert \wedge s_1[i_2] = v_1 = v_2) \vee$ $i_1 > i_2$	$(i_1 < i_2 < s_1.\texttt{size}() \wedge s_1.\texttt{get}(i_2 - 1) = s_1.\texttt{get}(i_2) = v_2) \vee$ $(i_1 = i_2 < s_1.\texttt{size}() \wedge s_1.\texttt{get}(i_2) = v_1 = v_2) \vee$ $i_1 > i_2$
	$r_2 = s_2.\texttt{size}()$	$false$	$false$
$r_1 = s_1.\texttt{get}(i_1)$	$s_2.\texttt{add\_at}(i_2, v_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1[i_1] = v_2) \vee$ $(i_1 > i_2 \wedge s_1[i_1 - 1] = s_1[i_1])$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = v_2) \vee$ $(i_1 > i_2 \wedge s_1.\texttt{get}(i_1 - 1) = s_1.\texttt{get}(i_1))$
	$r_2 = s_2.\texttt{get}(i_2)$	$true$	$true$
	$r_2 = s_2.\texttt{indexOf}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{lastIndexOf}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{remove\_at}(i_2)$	$i_1 < i_2 \vee$ $(\lvert s_1\rvert - 1 > i_1 = i_2 \wedge s_1[i_1] = s_1[i_1 + 1]) \vee$ $(\lvert s_1\rvert - 1 > i_1 > i_2 \wedge s_1[i_1] = s_1[i_1 + 1])$	$i_1 < i_2 \vee$ $(s_1.\texttt{size}() - 1 > i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = s_1.\texttt{get}(i_1 + 1)) \vee$ $(s_1.\texttt{size}() - 1 > i_1 > i_2 \wedge s_1.\texttt{get}(i_1) = s_1.\texttt{get}(i_1 + 1))$
	$s_2.\texttt{remove\_at}(i_2)$	$i_1 < i_2 \vee$ $(\lvert s_1\rvert - 1 > i_1 = i_2 \wedge s_1[i_1] = s_1[i_1 + 1]) \vee$ $(\lvert s_1\rvert - 1 > i_1 > i_2 \wedge s_1[i_1] = s_1[i_1 + 1])$	$i_1 < i_2 \vee$ $(s_1.\texttt{size}() - 1 > i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = s_1.\texttt{get}(i_1 + 1)) \vee$ $(s_1.\texttt{size}() - 1 > i_1 > i_2 \wedge s_1.\texttt{get}(i_1) = s_1.\texttt{get}(i_1 + 1))$
	$r_2 = s_2.\texttt{set}(i_2, v_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1[i_1] = v_2) \vee$ $i_1 > i_2$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = v_2) \vee$ $i_1 > i_2$
	$s_2.\texttt{set}(i_2, v_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1[i_1] = v_2) \vee$ $i_1 > i_2$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = v_2) \vee$ $i_1 > i_2$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$
$r_1 = s_1.\texttt{indexOf}(v_1)$	$s_2.\texttt{add\_at}(i_2, v_2)$	$(\neg(\exists i : s_1[i] = v_1) \wedge v_1 \neq v_2) \vee$ $(\exists i < i_2 : s_1[i] = v_1) \vee$ $(\neg(\exists i < i_2 : s_1[i] = v_1) \wedge s_1[i_2] = v_1 \wedge v_1 = v_2)$	$(s_1.\texttt{indexOf}(v_1) < 0 \wedge v_1 \neq v_2) \vee$ $0 \leq s_1.\texttt{indexOf}(v_1) < i_2 \vee$ $(s_1.\texttt{indexOf}(v_1) = i_2 \wedge v_1 = v_2)$
	$r_2 = s_2.\texttt{get}(i_2)$	$true$	$true$
	$r_2 = s_2.\texttt{indexOf}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{lastIndexOf}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{remove\_at}(i_2)$	$\neg(\exists i : s_1[i] = v_1) \vee$ $(\exists i < i_2 : s_1[i] = v_1) \vee$ $(\neg(\exists i < i_2 : s_1[i] = v_1) \wedge s_1[i_2] = v_1 \wedge i2 < \lvert s_1\rvert - 1 \wedge s_1[i_2 + 1] = v_1)$	$s_1.\texttt{indexOf}(v_1) < 0 \vee$ $0 \leq s_1.\texttt{indexOf}(v_1) < i_2 \vee$ $(s_1.\texttt{indexOf}(v_1) = i_2 \wedge i_2 < s_1.\texttt{size}() - 1 \wedge s_1.\texttt{get}(i_2 + 1) = v_1)$
	$s_2.\texttt{remove\_at}(i_2)$	$\neg(\exists i : s_1[i] = v_1) \vee$ $(\exists i < i_2 : s_1[i] = v_1) \vee$ $(\neg(\exists i < i_2 : s_1[i] = v_1) \wedge s_1[i_2] = v_1 \wedge i2 < \lvert s_1\rvert - 1 \wedge s_1[i_2 + 1] = v_1)$	$s_1.\texttt{indexOf}(v_1) < 0 \vee$ $0 \leq s_1.\texttt{indexOf}(v_1) < i_2 \vee$ $(s_1.\texttt{indexOf}(v_1) = i_2 \wedge i_2 < s_1.\texttt{size}() - 1 \wedge s_1.\texttt{get}(i_2 + 1) = v_1)$

Table 18: Before Commutativity Conditions on ArrayList (Cont.)

	$r_2 = s_2.\texttt{set}(i_2, v_2)$	$(\neg(\exists i : s_1[i] = v_1) \wedge v_1 \neq v_2) \vee$ $(\exists i < i_2 : s_1[i] = v_1) \vee$ $(\neg(\exists i < i_2 : s_1[i] = v_1) \wedge s_1[i_2] = v_1 \wedge v_1 = v_2) \vee$ $(\neg(\exists i \leq i_2 : s_1[i] = v_1) \wedge (\exists i > i_2 : s_1[i] = v_1) \wedge v_1 \neq v_2)$	$(s_1.\texttt{indexOf}(v_1) < 0 \wedge v_1 \neq v_2) \vee$ $0 \leq s_1.\texttt{indexOf}(v_1) < i_2 \vee$ $(s_1.\texttt{indexOf}(v_1) = i_2 \wedge v_1 = v_2) \vee$ $(s_1.\texttt{indexOf}(v_1) > i_2 \wedge v_1 \neq v_2)$
	$s_2.\texttt{set}(i_2, v_2)$	$(\neg(\exists i : s_1[i] = v_1) \wedge v_1 \neq v_2) \vee$ $(\exists i < i_2 : s_1[i] = v_1) \vee$ $(\neg(\exists i < i_2 : s_1[i] = v_1) \wedge s_1[i_2] = v_1 \wedge v_1 = v_2) \vee$ $(\neg(\exists i \leq i_2 : s_1[i] = v_1) \wedge (\exists i > i_2 : s_1[i] = v_1) \wedge v_1 \neq v_2)$	$(s_1.\texttt{indexOf}(v_1) < 0 \wedge v_1 \neq v_2) \vee$ $0 \leq s_1.\texttt{indexOf}(v_1) < i_2 \vee$ $(s_1.\texttt{indexOf}(v_1) = i_2 \wedge v_1 = v_2) \vee$ $(s_1.\texttt{indexOf}(v_1) > i_2 \wedge v_1 \neq v_2)$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$
$r_1 = s_1.\texttt{lastIndexOf}(v_1)$	$s_2.\texttt{add\_at}(i_2, v_2)$	$(\neg(\exists i : s_1[i] = v_1) \wedge v_1 \neq v_2) \vee$ $((\exists i < i_2 : s_1[i] = v_1) \wedge \neg(\exists i \geq i_2 : s_1[i] = v_1) \wedge v_1 \neq v_2)$	$(s_1.\texttt{lastIndexOf}(v_1) < 0 \wedge v_1 \neq v_2) \vee$ $(0 \leq s_1.\texttt{lastIndexOf}(v_1) < i_2 \wedge v_1 \neq v_2)$
	$r_2 = s_2.\texttt{get}(i_2)$	$true$	$true$
	$r_2 = s_2.\texttt{indexOf}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{lastIndexOf}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{remove\_at}(i_2)$	$\neg(\exists i : s_1[i] = v_1) \vee$ $((\exists i < i_2 : s_1[i] = v_1) \wedge \neg(\exists i \geq i_2 : s_1[i] = v_1))$	$s_1.\texttt{lastIndexOf}(v_1) < 0 \vee$ $0 \leq s_1.\texttt{lastIndexOf}(v_1) < i_2$
	$s_2.\texttt{remove\_at}(i_2)$	$\neg(\exists i : s_1[i] = v_1) \vee$ $((\exists i < i_2 : s_1[i] = v_1) \wedge \neg(\exists i \geq i_2 : s_1[i] = v_1))$	$s_1.\texttt{lastIndexOf}(v_1) < 0 \vee$ $0 \leq s_1.\texttt{lastIndexOf}(v_1) < i_2$
	$r_2 = s_2.\texttt{set}(i_2, v_2)$	$(\neg(\exists i : s_1[i] = v_1) \wedge v_1 \neq v_2) \vee$ $((\exists i < i_2 : s_1[i] = v_1) \wedge \neg(\exists i \geq i_2 : s_1[i] = v_1) \wedge v_1 \neq v_2) \vee$ $(s_1[i_2] = v_1 \wedge \neg(\exists i > i_2 : s_1[i] = v_1) \wedge v_1 = v_2) \vee$ $(\exists i > i_2 : s_1[i] = v_1)$	$(s_1.\texttt{lastIndexOf}(v_1) < 0 \wedge v_1 \neq v_2) \vee$ $(0 \leq s_1.\texttt{lastIndexOf}(v_1) < i_2 \wedge v_1 \neq v_2) \vee$ $(s_1.\texttt{lastIndexOf}(v_1) = i_2 \wedge v_1 = v_2) \vee$ $s_1.\texttt{lastIndexOf}(v_1) > i_2$
	$s_2.\texttt{set}(i_2, v_2)$	$(\neg(\exists i : s_1[i] = v_1) \wedge v_1 \neq v_2) \vee$ $((\exists i < i_2 : s_1[i] = v_1) \wedge \neg(\exists i \geq i_2 : s_1[i] = v_1) \wedge v_1 \neq v_2) \vee$ $(s_1[i_2] = v_1 \wedge \neg(\exists i > i_2 : s_1[i] = v_1) \wedge v_1 = v_2) \vee$ $(\exists i > i_2 : s_1[i] = v_1)$	$(s_1.\texttt{lastIndexOf}(v_1) < 0 \wedge v_1 \neq v_2) \vee$ $(0 \leq s_1.\texttt{lastIndexOf}(v_1) < i_2 \wedge v_1 \neq v_2) \vee$ $(s_1.\texttt{lastIndexOf}(v_1) = i_2 \wedge v_1 = v_2) \vee$ $s_1.\texttt{lastIndexOf}(v_1) > i_2$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$
$r_1 = s_1.\texttt{remove\_at}(i_1)$	$s_2.\texttt{add\_at}(i_2, v_2)$	$(i_1 < i_2 \wedge s_1[i_2] = v_2) \vee$ $(i_1 = i_2 \wedge s_1[i_1] = v_2) \vee$ $(i_1 > i_2 \wedge s_1[i_1 - 1] = s_1[i_1])$	$(i_1 < i_2 \wedge s_1.\texttt{get}(i_2) = v_2) \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = v_2) \vee$ $(i_1 > i_2 \wedge s_1.\texttt{get}(i_1 - 1) = s_1.\texttt{get}(i_1))$
	$r_2 = s_2.\texttt{get}(i_2)$	$(i_1 < i_2 \wedge s_1[i_2] = s_1[i_2 + 1]) \vee$ $(i_1 = i_2 \wedge s_1[i_1] = s_1[i_2 + 1]) \vee$ $i_1 > i_2$	$(i_1 < i_2 \wedge s_1.\texttt{get}(i_2) = s_1.\texttt{get}(i_2 + 1)) \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = s_1.\texttt{get}(i_2 + 1)) \vee$ $i_1 > i_2$
	$r_2 = s_2.\texttt{indexOf}(v_2)$	$\neg(\exists i : s_1[i] = v_2) \vee$ $(\exists i < i_1 : s_1[i] = v_2) \vee$ $(\neg(\exists i < i_1 : s_1[i] = v_2) \wedge s_1[i_1] = v_2 \wedge i_1 < \lvert s_1 \rvert - 1 \wedge s_1[i_1 + 1] = v_2)$	$s_1.\texttt{indexOf}(v_2) < 0 \vee$ $0 \leq s_1.\texttt{indexOf}(v_2) < i_1 \vee$ $(s_1.\texttt{indexOf}(v_2) = i_1 \wedge i_1 < s_1.\texttt{size}() - 1 \wedge s_1.\texttt{get}(i_1 + 1) = v_2)$
	$r_2 = s_2.\texttt{lastIndexOf}(v_2)$	$\neg(\exists i : s_1[i] = v_2) \vee$ $((\exists i < i_1 : s_1[i] = v_2) \wedge \neg(\exists i \geq i_1 : s_1[i] = v_2))$	$s_1.\texttt{lastIndexOf}(v_2) < 0 \vee$ $0 \leq s_1.\texttt{lastIndexOf}(v_2) < i_1$
	$r_2 = s_2.\texttt{remove\_at}(i_2)$	$(i_1 < i_2 \wedge s_1[i_2] = s_1[i_2 + 1]) \vee$ $(i_1 = i_2 \wedge s_1[i_1] = s_1[i_2 + 1]) \vee$ $(\lvert s_1 \rvert - 1 > i_1 > i_2 \wedge s_1[i_1] = s_1[i_1 + 1])$	$(i_1 < i_2 \wedge s_1.\texttt{get}(i_2) = s_1.\texttt{get}(i_2 + 1)) \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = s_1.\texttt{get}(i_2 + 1)) \vee$ $(s_1.\texttt{size}() - 1 > i_1 > i_2 \wedge s_1.\texttt{get}(i_1) = s_1.\texttt{get}(i_1 + 1))$
	$s_2.\texttt{remove\_at}(i_2)$	$(i_1 < i_2 \wedge s_1[i_2] = s_1[i_2 + 1]) \vee$ $(i_1 = i_2 \wedge s_1[i_1] = s_1[i_2 + 1]) \vee$ $(\lvert s_1 \rvert - 1 > i_1 > i_2 \wedge s_1[i_1] = s_1[i_1 + 1])$	$(i_1 < i_2 \wedge s_1.\texttt{get}(i_2) = s_1.\texttt{get}(i_2 + 1)) \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = s_1.\texttt{get}(i_2 + 1)) \vee$ $(s_1.\texttt{size}() - 1 > i_1 > i_2 \wedge s_1.\texttt{get}(i_1) = s_1.\texttt{get}(i_1 + 1))$
	$r_2 = s_2.\texttt{set}(i_2, v_2)$	$(i_1 < i_2 \wedge s_1[i_2] = s_1[i_2 + 1] = v_2) \vee$ $(i_1 = i_2 \wedge s_1[i_1] = s_1[i_2 + 1] = v_2) \vee$ $i_1 > i_2$	$(i_1 < i_2 \wedge s_1.\texttt{get}(i_2) = s_1.\texttt{get}(i_2 + 1) = v_2) \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = s_1.\texttt{get}(i_2 + 1) = v_2) \vee$ $i_1 > i_2$
	$s_2.\texttt{set}(i_2, v_2)$	$(i_1 < i_2 \wedge s_1[i_2] = s_1[i_2 + 1] = v_2) \vee$ $(i_1 = i_2 \wedge s_1[i_1] = s_1[i_2 + 1] = v_2) \vee$ $i_1 > i_2$	$(i_1 < i_2 \wedge s_1.\texttt{get}(i_2) = s_1.\texttt{get}(i_2 + 1) = v_2) \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = s_1.\texttt{get}(i_2 + 1) = v_2) \vee$ $i_1 > i_2$
	$r_2 = s_2.\texttt{size}()$	$false$	$false$
$s_1.\texttt{remove\_at}(i_1)$	$s_2.\texttt{add\_at}(i_2, v_2)$	$(i_1 < i_2 \wedge s_1[i_2] = v_2) \vee$ $(i_1 = i_2 \wedge s_1[i_1] = v_2) \vee$ $(i_1 > i_2 \wedge s_1[i_1 - 1] = s_1[i_1])$	$(i_1 < i_2 \wedge s_1.\texttt{get}(i_2) = v_2) \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = v_2) \vee$ $(i_1 > i_2 \wedge s_1.\texttt{get}(i_1 - 1) = s_1.\texttt{get}(i_1))$
	$r_2 = s_2.\texttt{get}(i_2)$	$(i_1 < i_2 \wedge s_1[i_2] = s_1[i_2 + 1]) \vee$ $(i_1 = i_2 \wedge s_1[i_1] = s_1[i_2 + 1]) \vee$ $i_1 > i_2$	$(i_1 < i_2 \wedge s_1.\texttt{get}(i_2) = s_1.\texttt{get}(i_2 + 1)) \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = s_1.\texttt{get}(i_2 + 1)) \vee$ $i_1 > i_2$
	$r_2 = s_2.\texttt{indexOf}(v_2)$	$\neg(\exists i : s_1[i] = v_2) \vee$ $(\exists i < i_1 : s_1[i] = v_2) \vee$ $(\neg(\exists i < i_1 : s_1[i] = v_2) \wedge s_1[i_1] = v_2 \wedge i_1 < \lvert s_1 \rvert - 1 \wedge s_1[i_1 + 1] = v_2)$	$s_1.\texttt{indexOf}(v_2) < 0 \vee$ $0 \leq s_1.\texttt{indexOf}(v_2) < i_1 \vee$ $(s_1.\texttt{indexOf}(v_2) = i_1 \wedge i_1 < s_1.\texttt{size}() - 1 \wedge s_1.\texttt{get}(i_1 + 1) = v_2)$

Table 18: Before Commutativity Conditions on ArrayList (Cont.)

	$r_2 = s_2.\texttt{lastIndexOf}(v_2)$	$\neg(\exists i : s_1[i] = v_2) \vee$ $((\exists i < i_1 : s_1[i] = v_2) \wedge \neg(\exists i \geq i_1 : s_1[i] = v_2))$	$s_1.\texttt{lastIndexOf}(v_2) < 0 \vee$ $0 \leq s_1.\texttt{lastIndexOf}(v_2) < i_1$
	$r_2 = s_2.\texttt{remove\_at}(i_2)$	$(i_1 < i_2 \wedge s_1[i_2] = s_1[i_2 + 1]) \vee$ $(i_1 = i_2 \wedge s_1[i_1] = s_1[i_2 + 1]) \vee$ $(\lvert s_1 \rvert - 1 > i_1 > i_2 \wedge s_1[i_1] = s_1[i_1 + 1])$	$(i_1 < i_2 \wedge s_1.\texttt{get}(i_2) = s_1.\texttt{get}(i_2 + 1)) \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = s_1.\texttt{get}(i_2 + 1)) \vee$ $(s_1.\texttt{size}() - 1 > i_1 > i_2 \wedge s_1.\texttt{get}(i_1) = s_1.\texttt{get}(i_1 + 1))$
	$s_2.\texttt{remove\_at}(i_2)$	$(i_1 < i_2 \wedge s_1[i_2] = s_1[i_2 + 1]) \vee$ $i_1 = i_2 \vee$ $(\lvert s_1 \rvert - 1 > i_1 > i_2 \wedge s_1[i_1] = s_1[i_1 + 1])$	$(i_1 < i_2 \wedge s_1.\texttt{get}(i_2) = s_1.\texttt{get}(i_2 + 1)) \vee$ $i_1 = i_2 \vee$ $(s_1.\texttt{size}() - 1 > i_1 > i_2 \wedge s_1.\texttt{get}(i_1) = s_1.\texttt{get}(i_1 + 1))$
	$r_2 = s_2.\texttt{set}(i_2, v_2)$	$(i_1 < i_2 \wedge s_1[i_2] = s_1[i_2 + 1] = v_2) \vee$ $(i_1 = i_2 \wedge s_1[i_1] = s_1[i_2 + 1] = v_2) \vee$ $i_1 > i_2$	$(i_1 < i_2 \wedge s_1.\texttt{get}(i_2) = s_1.\texttt{get}(i_2 + 1) = v_2) \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = s_1.\texttt{get}(i_2 + 1) = v_2) \vee$ $i_1 > i_2$
	$s_2.\texttt{set}(i_2, v_2)$	$(i_1 < i_2 \wedge s_1[i_2] = s_1[i_2 + 1] = v_2) \vee$ $(i_1 = i_2 \wedge s_1[i_2 + 1] = v_2) \vee$ $i_1 > i_2$	$(i_1 < i_2 \wedge s_1.\texttt{get}(i_2) = s_1.\texttt{get}(i_2 + 1) = v_2) \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_2 + 1) = v_2) \vee$ $i_1 > i_2$
	$r_2 = s_2.\texttt{size}()$	$false$	$false$
$r_1 = s_1.\texttt{set}(i_1, v_1)$	$s_2.\texttt{add\_at}(i_2, v_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1[i_1] = v_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_1[i_1 - 1] = s_1[i_1] = v_1)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = v_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_1.\texttt{get}(i_1 - 1) = s_1.\texttt{get}(i_1) = v_1)$
	$r_2 = s_2.\texttt{get}(i_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1[i_1] = v_1) \vee$ $i_1 > i_2$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = v_1) \vee$ $i_1 > i_2$
	$r_2 = s_2.\texttt{indexOf}(v_2)$	$(\neg(\exists i : s_1[i] = v_2) \wedge v_1 \neq v_2) \vee$ $(\exists i < i_1 : s_1[i] = v_2) \vee$ $(\neg(\exists i < i_1 : s_1[i] = v_2) \wedge s_1[i_1] = v_2 \wedge v_1 = v_2) \vee$ $(\neg(\exists i \leq i_1 : s_1[i] = v_2) \wedge (\exists i > i_1 : s_1[i] = v_2) \wedge v_1 \neq v_2)$	$(s_1.\texttt{indexOf}(v_2) < 0 \wedge v_1 \neq v_2) \vee$ $0 \leq s_1.\texttt{indexOf}(v_2) < i_1 \vee$ $(s_1.\texttt{indexOf}(v_2) = i_1 \wedge v_1 = v_2) \vee$ $(s_1.\texttt{indexOf}(v_2) > i_1 \wedge v_1 \neq v_2)$
	$r_2 = s_2.\texttt{lastIndexOf}(v_2)$	$(\neg(\exists i : s_1[i] = v_2) \wedge v_1 \neq v_2) \vee$ $((\exists i < i_1 : s_1[i] = v_2) \wedge \neg(\exists i \geq i_1 : s_1[i] = v_2) \wedge v_1 \neq v_2) \vee$ $(s_1[i_1] = v_2 \wedge \neg(\exists i > i_1 : s_1[i] = v_2) \wedge v_1 = v_2) \vee$ $(\exists i > i_1 : s_1[i] = v_2)$	$(s_1.\texttt{lastIndexOf}(v_2) < 0 \wedge v_1 \neq v_2) \vee$ $(0 \leq s_1.\texttt{lastIndexOf}(v_2) < i_1 \wedge v_1 \neq v_2) \vee$ $(s_1.\texttt{lastIndexOf}(v_2) = i_1 \wedge v_1 = v_2) \vee$ $s_1.\texttt{lastIndexOf}(v_2) > i_1$
	$r_2 = s_2.\texttt{remove\_at}(i_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 < \lvert s_1 \rvert - 1 \wedge s_1[i_1] = s_1[i_1 + 1] = v_1) \vee$ $(\lvert s_1 \rvert - 1 > i_1 > i_2 \wedge s_1[i_1] = s_1[i_1 + 1] = v_1)$	$i_1 < i_2 \vee$ $(i_1 = i_2 < s_1.\texttt{size}() - 1 \wedge s_1.\texttt{get}(i_1) = s_1.\texttt{get}(i_1 + 1) = v_1) \vee$ $(s_1.\texttt{size}() - 1 > i_1 > i_2 \wedge s_1.\texttt{get}(i_1) = s_1.\texttt{get}(i_1 + 1) = v_1)$
	$s_2.\texttt{remove\_at}(i_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 < \lvert s_1 \rvert - 1 \wedge s_1[i_1] = s_1[i_1 + 1] = v_1) \vee$ $(\lvert s_1 \rvert - 1 > i_1 > i_2 \wedge s_1[i_1] = s_1[i_1 + 1] = v_1)$	$i_1 < i_2 \vee$ $(i_1 = i_2 < s_1.\texttt{size}() - 1 \wedge s_1.\texttt{get}(i_1) = s_1.\texttt{get}(i_1 + 1) = v_1) \vee$ $(s_1.\texttt{size}() - 1 > i_1 > i_2 \wedge s_1.\texttt{get}(i_1) = s_1.\texttt{get}(i_1 + 1) = v_1)$
	$r_2 = s_2.\texttt{set}(i_2, v_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1[i_1] = v_1 = v_2) \vee$ $i_1 > i_2$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = v_1 = v_2) \vee$ $i_1 > i_2$
	$s_2.\texttt{set}(i_2, v_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1[i_1] = v_1 = v_2) \vee$ $i_1 > i_2$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = v_1 = v_2) \vee$ $i_1 > i_2$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$
$s_1.\texttt{set}(i_1, v_1)$	$s_2.\texttt{add\_at}(i_2, v_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1[i_1] = v_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_1[i_1 - 1] = s_1[i_1] = v_1)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = v_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_1.\texttt{get}(i_1 - 1) = s_1.\texttt{get}(i_1) = v_1)$
	$r_2 = s_2.\texttt{get}(i_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1[i_1] = v_1) \vee$ $i_1 > i_2$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = v_1) \vee$ $i_1 > i_2$
	$r_2 = s_2.\texttt{indexOf}(v_2)$	$(\neg(\exists i : s_1[i] = v_2) \wedge v_1 \neq v_2) \vee$ $(\exists i < i_1 : s_1[i] = v_2) \vee$ $(\neg(\exists i < i_1 : s_1[i] = v_2) \wedge s_1[i_1] = v_2 \wedge v_1 = v_2) \vee$ $(\neg(\exists i \leq i_1 : s_1[i] = v_2) \wedge (\exists i > i_1 : s_1[i] = v_2) \wedge v_1 \neq v_2)$	$(s_1.\texttt{indexOf}(v_2) < 0 \wedge v_1 \neq v_2) \vee$ $0 \leq s_1.\texttt{indexOf}(v_2) < i_1 \vee$ $(s_1.\texttt{indexOf}(v_2) = i_1 \wedge v_1 = v_2) \vee$ $(s_1.\texttt{indexOf}(v_2) > i_1 \wedge v_1 \neq v_2)$
	$r_2 = s_2.\texttt{lastIndexOf}(v_2)$	$(\neg(\exists i : s_1[i] = v_2) \wedge v_1 \neq v_2) \vee$ $((\exists i < i_1 : s_1[i] = v_2) \wedge \neg(\exists i \geq i_1 : s_1[i] = v_2) \wedge v_1 \neq v_2) \vee$ $(s_1[i_1] = v_2 \wedge \neg(\exists i > i_1 : s_1[i] = v_2) \wedge v_1 = v_2) \vee$ $(\exists i > i_1 : s_1[i] = v_2)$	$(s_1.\texttt{lastIndexOf}(v_2) < 0 \wedge v_1 \neq v_2) \vee$ $(0 \leq s_1.\texttt{lastIndexOf}(v_2) < i_1 \wedge v_1 \neq v_2) \vee$ $(s_1.\texttt{lastIndexOf}(v_2) = i_1 \wedge v_1 = v_2) \vee$ $s_1.\texttt{lastIndexOf}(v_2) > i_1$
	$r_2 = s_2.\texttt{remove\_at}(i_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 < \lvert s_1 \rvert - 1 \wedge s_1[i_1] = s_1[i_1 + 1] = v_1) \vee$ $(\lvert s_1 \rvert - 1 > i_1 > i_2 \wedge s_1[i_1] = s_1[i_1 + 1] = v_1)$	$i_1 < i_2 \vee$ $(i_1 = i_2 < s_1.\texttt{size}() - 1 \wedge s_1.\texttt{get}(i_1) = s_1.\texttt{get}(i_1 + 1) = v_1) \vee$ $(s_1.\texttt{size}() - 1 > i_1 > i_2 \wedge s_1.\texttt{get}(i_1) = s_1.\texttt{get}(i_1 + 1) = v_1)$
	$s_2.\texttt{remove\_at}(i_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 < \lvert s_1 \rvert - 1 \wedge s_1[i_1 + 1] = v_1) \vee$	$i_1 < i_2 \vee$ $(i_1 = i_2 < s_1.\texttt{size}() - 1 \wedge s_1.\texttt{get}(i_1 + 1) = v_1) \vee$

Table 18: Before Commutativity Conditions on ArrayList (Cont.)

		$(\lvert s_1\rvert - 1 > i_1 > i_2 \wedge s_1[i_1] = s_1[i_1 + 1] = v_1)$	$(s_1.\texttt{size}() - 1 > i_1 > i_2 \wedge s_1.\texttt{get}(i_1) = s_1.\texttt{get}(i_1 + 1) = v_1)$
	$r_2 = s_2.\texttt{set}(i_2, v_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1[i_1] = v_1 = v_2) \vee$ $i_1 > i_2$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = v_1 = v_2) \vee$ $i_1 > i_2$
	$s_2.\texttt{set}(i_2, v_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge v_1 = v_2) \vee$ $i_1 > i_2$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge v_1 = v_2) \vee$ $i_1 > i_2$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$
$r_1 = s_1.\texttt{size}(v_1)$	$s_2.\texttt{add\_at}(i_2, v_2)$	$false$	$false$
	$r_2 = s_2.\texttt{get}(i_2)$	$true$	$true$
	$r_2 = s_2.\texttt{indexOf}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{lastIndexOf}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{remove\_at}(i_2)$	$false$	$false$
	$s_2.\texttt{remove\_at}(i_2)$	$false$	$false$
	$r_2 = s_2.\texttt{set}(i_2, v_2)$	$true$	$true$
	$s_2.\texttt{set}(i_2, v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$

A.4.2 Between Commutativity Conditions

Table 19: Between Commutativity Conditions on ArrayList

Methods		Commutativity Condition	
$s_1.\texttt{add\_at}(i_1, v_1)$	$s_2.\texttt{add\_at}(i_2, v_2)$	$(i_1 < i_2 \leq \lvert s_2\rvert - 1 \wedge s_2[i_2] = v_2) \vee$ $(i_1 = i_2 \wedge v_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_2[i_1 - 1] = v_1)$	$(i_1 < i_2 \leq s_2.\texttt{size}() - 1 \wedge s_2.\texttt{get}(i_2) = v_2) \vee$ $(i_1 = i_2 \wedge v_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_2.\texttt{get}(i_1 - 1) = v_1)$
	$r_2 = s_2.\texttt{get}(i_2)$	$(i_1 < i_2 < \lvert s_2\rvert - 1 \wedge s_2[i_2] = s_2[i_2 + 1]) \vee$ $(i_1 = i_2 < \lvert s_2\rvert - 1 \wedge s_2[i_1 + 1] = v_1) \vee$ $i_1 > i_2$	$(i_1 < i_2 < s_2.\texttt{size}() - 1 \wedge s_2.\texttt{get}(i_2) = s_2.\texttt{get}(i_2 + 1)) \vee$ $(i_1 = i_2 < s_2.\texttt{size}() - 1 \wedge s_2.\texttt{get}(i_1 + 1) = v_1) \vee$ $i_1 > i_2$
	$r_2 = s_2.\texttt{indexOf}(v_2)$	$\neg(\exists i : s_2[i] = v_2) \vee$ $(\exists i < i_1 : s_2[i] = v_2) \vee$ $(\neg(\exists i < i_1 : s_2[i] = v_2) \wedge s_2[i_1] = v_2 \wedge s_2[i_1 + 1] = v_2)$	$s_2.\texttt{indexOf}(v_2) < 0 \vee$ $0 \leq s_2.\texttt{indexOf}(v_2) < i_1 \vee$ $(s_2.\texttt{indexOf}(v_2) = i_1 \wedge s_2.\texttt{get}(i_1 + 1) = v_2)$
	$r_2 = s_2.\texttt{lastIndexOf}(v_2)$	$\neg(\exists i : s_2[i] = v_2) \vee$ $((\exists i < i_1 : s_2[i] = v_2) \wedge \neg(\exists i \geq i_1 : s_2[i] = v_2))$	$s_2.\texttt{lastIndexOf}(v_2) < 0 \vee$ $0 \leq s_2.\texttt{lastIndexOf}(v_2) < i_1$
	$r_2 = s_2.\texttt{remove\_at}(i_2)$	$(i_1 < i_2 < \lvert s_2\rvert - 1 \wedge s_2[i_2] = s_2[i_2 + 1]) \vee$ $(\lvert s_2\rvert - 2 \geq i_1 = i_2 \wedge s_2[i_1 + 1] = v_1) \vee$ $(\lvert s_2\rvert - 2 \geq i_1 > i_2 \wedge s_2[i_1 + 1] = v_1)$	$(i_1 < i_2 < s_2.\texttt{size}() - 1 \wedge s_2.\texttt{get}(i_2) = s_2.\texttt{get}(i_2 + 1)) \vee$ $(s_2.\texttt{size}() - 2 \geq i_1 = i_2 \wedge s_2.\texttt{get}(i_1 + 1) = v_1) \vee$ $(s_2.\texttt{size}() - 2 \geq i_1 > i_2 \wedge s_2.\texttt{get}(i_1 + 1) = v_1)$
	$s_2.\texttt{remove\_at}(i_2)$	$(i_1 < i_2 < \lvert s_2\rvert - 1 \wedge s_2[i_2] = s_2[i_2 + 1]) \vee$ $(\lvert s_2\rvert - 2 \geq i_1 = i_2 \wedge s_2[i_1 + 1] = v_1) \vee$ $(\lvert s_2\rvert - 2 \geq i_1 > i_2 \wedge s_2[i_1 + 1] = v_1)$	$(i_1 < i_2 < s_2.\texttt{size}() - 1 \wedge s_2.\texttt{get}(i_2) = s_2.\texttt{get}(i_2 + 1)) \vee$ $(s_2.\texttt{size}() - 2 \geq i_1 = i_2 \wedge s_2.\texttt{get}(i_1 + 1) = v_1) \vee$ $(s_2.\texttt{size}() - 2 \geq i_1 > i_2 \wedge s_2.\texttt{get}(i_1 + 1) = v_1)$
	$r_2 = s_2.\texttt{set}(i_2, v_2)$	$(i_1 < i_2 < \lvert s_2\rvert - 1 \wedge s_2[i_2] = s_2[i_2 + 1] = v_2) \vee$ $(i_1 = i_2 < \lvert s_2\rvert - 1 \wedge s_2[i_2 + 1] = v_1 = v_2) \vee$ $i_1 > i_2$	$(i_1 < i_2 < s_2.\texttt{size}() - 1 \wedge s_2.\texttt{get}(i_2) = s_2.\texttt{get}(i_2 + 1) = v_2) \vee$ $(i_1 = i_2 < s_2.\texttt{size}() - 1 \wedge s_2.\texttt{get}(i_2 + 1) = v_1 = v_2) \vee$ $i_1 > i_2$
	$s_2.\texttt{set}(i_2, v_2)$	$(i_1 < i_2 < \lvert s_2\rvert - 1 \wedge s_2[i_2] = s_2[i_2 + 1] = v_2) \vee$ $(i_1 = i_2 < \lvert s_2\rvert - 1 \wedge s_2[i_2 + 1] = v_1 = v_2) \vee$ $i_1 > i_2$	$(i_1 < i_2 < s_2.\texttt{size}() - 1 \wedge s_2.\texttt{get}(i_2) = s_2.\texttt{get}(i_2 + 1) = v_2) \vee$ $(i_1 = i_2 < s_2.\texttt{size}() - 1 \wedge s_2.\texttt{get}(i_2 + 1) = v_1 = v_2) \vee$ $i_1 > i_2$
	$r_2 = s_2.\texttt{size}()$	$false$	$false$
$r_1 = s_1.\texttt{get}(i_1)$	$s_2.\texttt{add\_at}(i_2, v_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge r_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_2[i_1 - 1] = r_1)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge r_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_2.\texttt{get}(i_1 - 1) = r_1)$
	$r_2 = s_2.\texttt{get}(i_2)$	$true$	$true$
	$r_2 = s_2.\texttt{indexOf}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{lastIndexOf}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{remove\_at}(i_2)$	$i_1 < i_2 \vee$ $(\lvert s_2\rvert - 1 > i_1 = i_2 \wedge r_1 = s_2[i_1 + 1]) \vee$ $(\lvert s_2\rvert - 1 > i_1 > i_2 \wedge r_1 = s_2[i_1 + 1])$	$i_1 < i_2 \vee$ $(s_2.\texttt{size}() - 1 > i_1 = i_2 \wedge r_1 = s_2.\texttt{get}(i_1 + 1)) \vee$ $(s_2.\texttt{size}() - 1 > i_1 > i_2 \wedge r_1 = s_2.\texttt{get}(i_1 + 1))$
	$s_2.\texttt{remove\_at}(i_2)$	$i_1 < i_2 \vee$ $(\lvert s_2\rvert - 1 > i_1 = i_2 \wedge r_1 = s_2[i_1 + 1]) \vee$	$i_1 < i_2 \vee$ $(s_2.\texttt{size}() - 1 > i_1 = i_2 \wedge r_1 = s_2.\texttt{get}(i_1 + 1)) \vee$

Table 19: Between Commutativity Conditions on ArrayList (Cont.)

		$(\lvert s_2\rvert - 1 > i_1 > i_2 \wedge r_1 = s_2[i_1 + 1])$	$(s_2.\texttt{size}() - 1 > i_1 > i_2 \wedge r_1 = s_2.\texttt{get}(i_1 + 1))$
	$r_2 = s_2.\texttt{set}(i_2, v_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge r_1 = v_2) \vee$ $i_1 > i_2$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge r_1 = v_2) \vee$ $i_1 > i_2$
	$s_2.\texttt{set}(i_2, v_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge r_1 = v_2) \vee$ $i_1 > i_2$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge r_1 = v_2) \vee$ $i_1 > i_2$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$
$r_1 = s_1.\texttt{indexOf}(v_1)$	$s_2.\texttt{add\_at}(i_2, v_2)$	$(r_1 < 0 \wedge v_1 \neq v_2) \vee$ $0 \leq r_1 < i_2 \vee$ $(r_1 = i_2 \wedge v_1 = v_2)$	$(r_1 < 0 \wedge v_1 \neq v_2) \vee$ $0 \leq r_1 < i_2 \vee$ $(r_1 = i_2 \wedge v_1 = v_2)$
	$r_2 = s_2.\texttt{get}(i_2)$	$true$	$true$
	$r_2 = s_2.\texttt{indexOf}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{lastIndexOf}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{remove\_at}(i_2)$	$r_1 < 0 \vee$ $0 \leq r_1 < i_2 \vee$ $(r_1 = i_2 \wedge i_2 < \lvert s_2\rvert - 1 \wedge s_2[i_2 + 1] = v_1)$	$r_1 < 0 \vee$ $0 \leq r_1 < i_2 \vee$ $(r_1 = i_2 \wedge i_2 < s_2.\texttt{size}() - 1 \wedge s_2.\texttt{get}(i_2 + 1) = v_1)$
	$s_2.\texttt{remove\_at}(i_2)$	$r_1 < 0 \vee$ $0 \leq r_1 < i_2 \vee$ $(r_1 = i_2 \wedge i_2 < \lvert s_2\rvert - 1 \wedge s_2[i_2 + 1] = v_1)$	$r_1 < 0 \vee$ $0 \leq r_1 < i_2 \vee$ $(r_1 = i_2 \wedge i_2 < s_2.\texttt{size}() - 1 \wedge s_2.\texttt{get}(i_2 + 1) = v_1)$
	$r_2 = s_2.\texttt{set}(i_2, v_2)$	$(r_1 < 0 \wedge v_1 \neq v_2) \vee$ $0 \leq r_1 < i_2 \vee$ $(r_1 = i_2 \wedge v_1 = v_2) \vee$ $(r_1 > i_2 \wedge v_1 \neq v_2)$	$(r_1 < 0 \wedge v_1 \neq v_2) \vee$ $0 \leq r_1 < i_2 \vee$ $(r_1 = i_2 \wedge v_1 = v_2) \vee$ $(r_1 > i_2 \wedge v_1 \neq v_2)$
	$s_2.\texttt{set}(i_2, v_2)$	$(r_1 < 0 \wedge v_1 \neq v_2) \vee$ $0 \leq r_1 < i_2 \vee$ $(r_1 = i_2 \wedge v_1 = v_2) \vee$ $(r_1 > i_2 \wedge v_1 \neq v_2)$	$(r_1 < 0 \wedge v_1 \neq v_2) \vee$ $0 \leq r_1 < i_2 \vee$ $(r_1 = i_2 \wedge v_1 = v_2) \vee$ $(r_1 > i_2 \wedge v_1 \neq v_2)$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$
$r_1 = s_1.\texttt{lastIndexOf}(v_1)$	$s_2.\texttt{add\_at}(i_2, v_2)$	$(r_1 < 0 \wedge v_1 \neq v_2) \vee$ $(0 \leq r_1 < i_2 \wedge v_1 \neq v_2)$	$(r_1 < 0 \wedge v_1 \neq v_2) \vee$ $(0 \leq r_1 < i_2 \wedge v_1 \neq v_2)$
	$r_2 = s_2.\texttt{get}(i_2)$	$true$	$true$
	$r_2 = s_2.\texttt{indexOf}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{lastIndexOf}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{remove\_at}(i_2)$	$r_1 < 0 \vee$ $0 \leq r_1 < i_2$	$r_1 < 0 \vee$ $0 \leq r_1 < i_2$
	$s_2.\texttt{remove\_at}(i_2)$	$r_1 < 0 \vee$ $0 \leq r_1 < i_2$	$r_1 < 0 \vee$ $0 \leq r_1 < i_2$
	$r_2 = s_2.\texttt{set}(i_2, v_2)$	$(r_1 < 0 \wedge v_1 \neq v_2) \vee$ $(0 \leq r_1 < i_2 \wedge v_1 \neq v_2) \vee$ $(r_1 = i_2 \wedge v_1 = v_2) \vee$ $r_1 > i_2$	$(r_1 < 0 \wedge v_1 \neq v_2) \vee$ $(0 \leq r_1 < i_2 \wedge v_1 \neq v_2) \vee$ $(r_1 = i_2 \wedge v_1 = v_2) \vee$ $r_1 > i_2$
	$s_2.\texttt{set}(i_2, v_2)$	$(r_1 < 0 \wedge v_1 \neq v_2) \vee$ $(0 \leq r_1 < i_2 \wedge v_1 \neq v_2) \vee$ $(r_1 = i_2 \wedge v_1 = v_2) \vee$ $r_1 > i_2$	$(r_1 < 0 \wedge v_1 \neq v_2) \vee$ $(0 \leq r_1 < i_2 \wedge v_1 \neq v_2) \vee$ $(r_1 = i_2 \wedge v_1 = v_2) \vee$ $r_1 > i_2$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$
$r_1 = s_1.\texttt{remove\_at}(i_1)$	$s_2.\texttt{add\_at}(i_2, v_2)$	$(i_1 < i_2 \wedge s_2[i_2 - 1] = v_2) \vee$ $(i_1 = i_2 \wedge r_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_2[i_1 - 1] = r_1)$	$(i_1 < i_2 \wedge s_2.\texttt{get}(i_2 - 1) = v_2) \vee$ $(i_1 = i_2 \wedge r_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_2.\texttt{get}(i_1 - 1) = r_1)$
	$r_2 = s_2.\texttt{get}(i_2)$	$(i_1 < i_2 \wedge s_2[i_2 - 1] = s_2[i_2]) \vee$ $(i_1 = i_2 \wedge r_1 = s_2[i_2]) \vee$ $i_1 > i_2$	$(i_1 < i_2 \wedge s_2.\texttt{get}(i_2 - 1) = s_2.\texttt{get}(i_2)) \vee$ $(i_1 = i_2 \wedge r_1 = s_2.\texttt{get}(i_2)) \vee$ $i_1 > i_2$
	$r_2 = s_2.\texttt{indexOf}(v_2)$	$(\neg(\exists i : s_2[i] = v_2) \wedge r_1 \neq v_2) \vee$ $(\exists i < i_1 : s_2[i] = v_2) \vee$ $(\neg(\exists i < i_1 : s_2[i] = v_2) \wedge s_2[i_1] = v_2 \wedge r_1 = v_2 \wedge i_1 < \lvert s_2\rvert)$	$(s_2.\texttt{indexOf}(v_2) < 0 \wedge r_1 \neq v_2) \vee$ $0 \leq s_2.\texttt{indexOf}(v_2) < i_1 \vee$ $(s_2.\texttt{indexOf}(v_2) = i_1 \wedge r_1 = v_2 \wedge i_1 < s_2.\texttt{size}())$
	$r_2 = s_2.\texttt{lastIndexOf}(v_2)$	$(\neg(\exists i : s_2[i] = v_2) \wedge r_1 \neq v_2) \vee$ $((\exists i < i_1 : s_2[i] = v_2) \wedge \neg(\exists i \geq i_1 : s_2[i] = v_2) \wedge r_1 \neq v_2)$	$(s_2.\texttt{lastIndexOf}(v_2) < 0 \wedge r_1 \neq v_2) \vee$ $(0 \leq s_2.\texttt{lastIndexOf}(v_2) < i_1 \wedge r_1 \neq v_2)$
	$r_2 = s_2.\texttt{remove\_at}(i_2)$	$(i_1 < i_2 \wedge s_2[i_2 - 1] = s_2[i_2]) \vee$ $(i_1 = i_2 \wedge r_1 = s_2[i_2]) \vee$	$(i_1 < i_2 \wedge s_2.\texttt{get}(i_2 - 1) = s_2.\texttt{get}(i_2)) \vee$ $(i_1 = i_2 \wedge r_1 = s_2.\texttt{get}(i_2)) \vee$

Table 19: Between Commutativity Conditions on ArrayList (Cont.)

		$(\lvert s_2\rvert > i_1 > i_2 \wedge r_1 = s_2[i_1])$	$(s_2.\texttt{size}() > i_1 > i_2 \wedge r_1 = s_2.\texttt{get}(i_1))$
	$s_2.\texttt{remove\_at}(i_2)$	$(i_1 < i_2 \wedge s_2[i_2 - 1] = s_2[i_2]) \vee$ $(i_1 = i_2 \wedge r_1 = s_2[i_2]) \vee$ $(\lvert s_2\rvert > i_1 > i_2 \wedge r_1 = s_2[i_1])$	$(i_1 < i_2 \wedge s_2.\texttt{get}(i_2 - 1) = s_2.\texttt{get}(i_2)) \vee$ $(i_1 = i_2 \wedge r_1 = s_2.\texttt{get}(i_2)) \vee$ $(s_2.\texttt{size}() > i_1 > i_2 \wedge r_1 = s_2.\texttt{get}(i_1))$
	$r_2 = s_2.\texttt{set}(i_2, v_2)$	$(i_1 < i_2 \wedge s_2[i_2 - 1] = s_2[i_2] = v_2) \vee$ $(i_1 = i_2 \wedge r_1 = s_2[i_2] = v_2) \vee$ $i_1 > i_2$	$(i_1 < i_2 \wedge s_2.\texttt{get}(i_2 - 1) = a.\texttt{get}(i_2) = v_2) \vee$ $(i_1 = i_2 \wedge r_1 = s_2.\texttt{get}(i_2) = v_2) \vee$ $i_1 > i_2$
	$s_2.\texttt{set}(i_2, v_2)$	$(i_1 < i_2 \wedge s_2[i_2 - 1] = s_2[i_2] = v_2) \vee$ $(i_1 = i_2 \wedge r_1 = s_2[i_2] = v_2) \vee$ $i_1 > i_2$	$(i_1 < i_2 \wedge s_2.\texttt{get}(i_2 - 1) = a.\texttt{get}(i_2) = v_2) \vee$ $(i_1 = i_2 \wedge r_1 = s_2.\texttt{get}(i_2) = v_2) \vee$ $i_1 > i_2$
	$r_2 = s_2.\texttt{size}()$	$false$	$false$
$s_1.\texttt{remove\_at}(i_1)$	$s_2.\texttt{add\_at}(i_2, v_2)$	$(i_1 < i_2 \wedge s_2[i_2 - 1] = v_2) \vee$ $(i_1 = i_2 \wedge s_1[i_1] = v_2) \vee$ $(i_1 > i_2 \wedge s_2[i_1 - 1] = s_1[i_1])$	$(i_1 < i_2 \wedge s_2.\texttt{get}(i_2 - 1) = v_2) \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = v_2) \vee$ $(i_1 > i_2 \wedge s_2.\texttt{get}(i_1 - 1) = s_1.\texttt{get}(i_1))$
	$r_2 = s_2.\texttt{get}(i_2)$	$(i_1 < i_2 \wedge s_2[i_2 - 1] = s_2[i_2]) \vee$ $(i_1 = i_2 \wedge s_1[i_1] = s_2[i_2]) \vee$ $i_1 > i_2$	$(i_1 < i_2 \wedge s_2.\texttt{get}(i_2 - 1) = s_2.\texttt{get}(i_2)) \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i1) = s_2.\texttt{get}(i_2)) \vee$ $i_1 > i_2$
	$r_2 = s_2.\texttt{indexOf}(v_2)$	$(\neg(\exists i : s_2[i] = v_2) \wedge s_1[i_1] \neq v_2) \vee$ $(\exists i < i_1 : s_2[i] = v_2) \vee$ $(\neg(\exists i < i_1 : s_2[i] = v_2) \wedge s_2[i_1] = v_2 \wedge s_1[i_1] = v_2 \wedge i_1 < \lvert s_2\rvert)$	$(s_2.\texttt{indexOf}(v_2) < 0 \wedge s_1.\texttt{get}(i_1) \neq v_2) \vee$ $0 \leq s_2.\texttt{indexOf}(v_2) < i_1 \vee$ $(s_2.\texttt{indexOf}(v_2) = i_1 \wedge s_1.\texttt{get}(i_1) = v_2 \wedge i_1 < s_2.\texttt{size}())$
	$r_2 = s_2.\texttt{lastIndexOf}(v_2)$	$(\neg(\exists i : s_2[i] = v_2) \wedge s_1[i_1] \neq v_2) \vee$ $((\exists i < i_1 : s_2[i] = v_2) \wedge \neg(\exists i \geq i_1 : s_2[i] = v_2) \wedge s_1[i_1] \neq v_2)$	$(s_2.\texttt{lastIndexOf}(v_2) < 0 \wedge s_1.\texttt{get}(i_1) \neq v_2) \vee$ $(0 \leq s_2.\texttt{lastIndexOf}(v_2) < i_1 \wedge s_1.\texttt{get}(i_1) \neq v_2)$
	$r_2 = s_2.\texttt{remove\_at}(i_2)$	$(i_1 < i_2 \wedge s_2[i_2 - 1] = s_2[i_2]) \vee$ $(i_1 = i_2 \wedge s_1[i_1] = s_2[i_2]) \vee$ $(\lvert s_2\rvert > i_1 > i_2 \wedge s_1[i_1] = s_2[i_1])$	$(i_1 < i_2 \wedge s_2.\texttt{get}(i_2 - 1) = s_2.\texttt{get}(i_2)) \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = s_2.\texttt{get}(i_2)) \vee$ $(s_2.\texttt{size}() > i_1 > i_2 \wedge s_1.\texttt{get}(i_1) = s_2.\texttt{get}(i_1))$
	$s_2.\texttt{remove\_at}(i_2)$	$(i_1 < i_2 \wedge s_2[i_2 - 1] = s_2[i_2]) \vee$ $i_1 = i_2 \vee$ $(\lvert s_2\rvert > i_1 > i_2 \wedge s_1[i_1] = s_2[i_1])$	$(i_1 < i_2 \wedge s_2.\texttt{get}(i_2 - 1) = s_2.\texttt{get}(i_2)) \vee$ $i_1 = i_2 \vee$ $(s_2.\texttt{size}() > i_1 > i_2 \wedge s_1.\texttt{get}(i_1) = s_2.\texttt{get}(i_1))$
	$r_2 = s_2.\texttt{set}(i_2, v_2)$	$(i_1 < i_2 \wedge s_2[i_2 - 1] = s_2[i_2] = v_2) \vee$ $(i_1 = i_2 \wedge s_1[i_1] = s_2[i_2] = v_2) \vee$ $i_1 > i_2$	$(i_1 < i_2 \wedge s_2.\texttt{get}(i_2 - 1) = a.\texttt{get}(i_2) = v_2) \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = s_2.\texttt{get}(i_2) = v_2) \vee$ $i_1 > i_2$
	$s_2.\texttt{set}(i_2, v_2)$	$(i_1 < i_2 \wedge s_2[i_2 - 1] = s_2[i_2] = v_2) \vee$ $(i_1 = i_2 \wedge s_2[i_2] = v_2) \vee$ $i_1 > i_2$	$(i_1 < i_2 \wedge s_2.\texttt{get}(i_2 - 1) = s_2.\texttt{get}(i_2) = v_2) \vee$ $(i_1 = i_2 \wedge s_2.\texttt{get}(i_2) = v_2) \vee$ $i_1 > i_2$
	$r_2 = s_2.\texttt{size}()$	$false$	$false$
$r_1 = s_1.\texttt{set}(i_1, v_1)$	$s_2.\texttt{add\_at}(i_2, v_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge r_1 = v_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_2[i_1 - 1] = r_1 = v_1)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge r_1 = v_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_2.\texttt{get}(i_1 - 1) = r_1 = v_1)$
	$r_2 = s_2.\texttt{get}(i_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge r_1 = v_1) \vee$ $i_1 > i_2$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge r_1 = v_1) \vee$ $i_1 > i_2$
	$r_2 = s_2.\texttt{indexOf}(v_2)$	$(\neg(\exists i : s_2[i] = v_2) \wedge r_1 \neq v_2) \vee$ $(\exists i < i_1 : s_2[i] = v_2) \vee$ $(\neg(\exists i < i_1 : s_2[i] = v_2) \wedge s_2[i_1] = v_2 \wedge r_1 = v_2) \vee$ $(\neg(\exists i \leq i_1 : s_2[i] = v_2) \wedge (\exists i > i_1 : s_2[i] = v_2) \wedge r_1 \neq v_2)$	$(s_2.\texttt{indexOf}(v_2) < 0 \wedge r_1 \neq v_2) \vee$ $0 \leq s_2.\texttt{indexOf}(v_2) < i_1 \vee$ $(s_2.\texttt{indexOf}(v_2) = i_1 \wedge r_1 = v_2) \vee$ $(s_2.\texttt{indexOf}(v_2) > i_1 \wedge r_1 \neq v_2)$
	$r_2 = s_2.\texttt{lastIndexOf}(v_2)$	$(\neg(\exists i : s_2[i] = v_2) \wedge r_1 \neq v_2) \vee$ $((\exists i < i_1 : s_2[i] = v_2) \wedge \neg(\exists i \geq i_1 : s_2[i] = v_2) \wedge r_1 \neq v_2) \vee$ $(s_2[i_1] = v_2 \wedge \neg(\exists i > i_1 : s_2[i] = v_2) \wedge r_1 = v_2) \vee$ $(\exists i > i_1 : s_2[i] = v_2)$	$(s_2.\texttt{lastIndexOf}(v_2) < 0 \wedge r_1 \neq v_2) \vee$ $(0 \leq s_2.\texttt{lastIndexOf}(v_2) < i_1 \wedge r_1 \neq v_2) \vee$ $(s_2.\texttt{lastIndexOf}(v_2) = i_1 \wedge r_1 = v_2) \vee$ $s_2.\texttt{lastIndexOf}(v_2) > i_1$
	$r_2 = s_2.\texttt{remove\_at}(i_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 < \lvert s_2\rvert - 1 \wedge r_1 = s_2[i_1 + 1] = v_1) \vee$ $(\lvert s_2\rvert - 1 > i_1 > i_2 \wedge r_1 = s_2[i_1 + 1] = v_1)$	$i_1 < i_2 \vee$ $(i_1 = i_2 < s_2.\texttt{size}() - 1 \wedge r_1 = s_2.\texttt{get}(i_1 + 1) = v_1) \vee$ $(s_2.\texttt{size}() - 1 > i_1 > i_2 \wedge r_1 = s_2.\texttt{get}(i_1 + 1) = v_1)$
	$s_2.\texttt{remove\_at}(i_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 < \lvert s_2\rvert - 1 \wedge r_1 = s_2[i_1 + 1] = v_1) \vee$ $(\lvert s_2\rvert - 1 > i_1 > i_2 \wedge r_1 = s_2[i_1 + 1] = v_1)$	$i_1 < i_2 \vee$ $(i_1 = i_2 < s_2.\texttt{size}() - 1 \wedge r_1 = s_2.\texttt{get}(i_1 + 1) = v_1) \vee$ $(s_2.\texttt{size}() - 1 > i_1 > i_2 \wedge r_1 = s_2.\texttt{get}(i_1 + 1) = v_1)$
	$r_2 = s_2.\texttt{set}(i_2, v_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge r_1 = v_1 = v_2) \vee$ $i_1 > i_2$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge r_1 = v_1 = v_2) \vee$ $i_1 > i_2$
	$s_2.\texttt{set}(i_2, v_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge r_1 = v_1 = v_2) \vee$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge r_1 = v_1 = v_2) \vee$

Table 19: Between Commutativity Conditions on ArrayList (Cont.)

		$i_1 > i_2$	$i_1 > i_2$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$
$s_1.\texttt{set}(i_1, v_1)$	$s_2.\texttt{add\_at}(i_2, v_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1[i_1] = v_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_2[i_1 - 1] = s_1[i_1] = v_1)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = v_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_2.\texttt{get}(i_1 - 1) = s_1.\texttt{get}(i_1) = v_1)$
	$r_2 = s_2.\texttt{get}(i_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1[i_1] = v_1) \vee$ $i_1 > i_2$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = v_1) \vee$ $i_1 > i_2$
	$r_2 = s_2.\texttt{indexOf}(v_2)$	$(\neg(\exists i : s_2[i] = v_2) \wedge s_1[i_1] \neq v_2) \vee$ $(\exists i < i_1 : s_2[i] = v_2) \vee$ $(\neg(\exists i < i_1 : s_2[i] = v_2) \wedge s_2[i_1] = v_2 \wedge s_1[i_1] = v_2) \vee$ $(\neg(\exists i \leq i_1 : s_2[i] = v_2) \wedge (\exists i > i_1 : s_2[i] = v_2) \wedge s_1[i_1] \neq v_2)$	$(s_2.\texttt{indexOf}(v_2) < 0 \wedge s_1.\texttt{get}(i_1) \neq v_2) \vee$ $0 \leq s_2.\texttt{indexOf}(v_2) < i_1 \vee$ $(s_2.\texttt{indexOf}(v_2) = i_1 \wedge s_1.\texttt{get}(i_1) = v_2) \vee$ $(s_2.\texttt{indexOf}(v_2) > i_1 \wedge s_1.\texttt{get}(i_1) \neq v_2)$
	$r_2 = s_2.\texttt{lastIndexOf}(v_2)$	$(\neg(\exists i : s_2[i] = v_2) \wedge s_1[i_1] \neq v_2) \vee$ $((\exists i < i_1 : s_2[i] = v_2) \wedge \neg(\exists i \geq i_1 : s_2[i] = v_2) \wedge s_1[i_1] \neq v_2) \vee$ $(s_2[i_1] = v_2 \wedge \neg(\exists i > i_1 : s_2[i] = v_2) \wedge s_1[i_1] = v_2) \vee$ $(\exists i > i_1 : s_2[i] = v_2)$	$(s_2.\texttt{lastIndexOf}(v_2) < 0 \wedge s_1.\texttt{get}(i_1) \neq v_2) \vee$ $(0 \leq s_2.\texttt{lastIndexOf}(v_2) < i_1 \wedge s_1.\texttt{get}(i_1) \neq v_2) \vee$ $(s_2.\texttt{lastIndexOf}(v_2) = i_1 \wedge s_1.\texttt{get}(i_1) = v_2) \vee$ $s_2.\texttt{lastIndexOf}(v_2) > i_1$
	$r_2 = s_2.\texttt{remove\_at}(i_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 < \lvert s_2 \rvert - 1 \wedge s_1[i_1] = s_2[i_1 + 1] = v_1) \vee$ $(\lvert s_2 \rvert - 1 > i_1 > i_2 \wedge s_1[i_1] = s_2[i_1 + 1] = v_1)$	$i_1 < i_2 \vee$ $(i_1 = i_2 < s_2.\texttt{size}() - 1 \wedge s_1.\texttt{get}(i_1) = s_2.\texttt{get}(i_1 + 1) = v_1) \vee$ $(s_2.\texttt{size}() - 1 > i_1 > i_2 \wedge s_1.\texttt{get}(i_1) = s_2.\texttt{get}(i_1 + 1) = v_1)$
	$s_2.\texttt{remove\_at}(i_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 < \lvert s_2 \rvert - 1 \wedge s_2[i_1 + 1] = v_1) \vee$ $(\lvert s_2 \rvert - 1 > i_1 > i_2 \wedge s_1[i_1] = s_2[i_1 + 1] = v_1)$	$i_1 < i_2 \vee$ $(i_1 = i_2 < s_2.\texttt{size}() - 1 \wedge s_2.\texttt{get}(i_1 + 1) = v_1) \vee$ $(s_2.\texttt{size}() - 1 > i_1 > i_2 \wedge s_1.\texttt{get}(i_1) = s_2.\texttt{get}(i_1 + 1) = v_1)$
	$r_2 = s_2.\texttt{set}(i_2, v_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1[i_1] = v_1 = v_2) \vee$ $i_1 > i_2$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = v_1 = v_2) \vee$ $i_1 > i_2$
	$s_2.\texttt{set}(i_2, v_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge v_1 = v_2) \vee$ $i_1 > i_2$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge v_1 = v_2) \vee$ $i_1 > i_2$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$
$r_1 = s_1.\texttt{size}(v_1)$	$s_2.\texttt{add\_at}(i_2, v_2)$	$false$	$false$
	$r_2 = s_2.\texttt{get}(i_2)$	$true$	$true$
	$r_2 = s_2.\texttt{indexOf}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{lastIndexOf}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{remove\_at}(i_2)$	$false$	$false$
	$s_2.\texttt{remove\_at}(i_2)$	$false$	$false$
	$r_2 = s_2.\texttt{set}(i_2, v_2)$	$true$	$true$
	$s_2.\texttt{set}(i_2, v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$

A.4.3 After Commutativity Conditions

Table 20: After Commutativity Conditions on ArrayList

Methods		Commutativity Condition	
$s_1.\texttt{add\_at}(i_1, v_1)$	$s_2.\texttt{add\_at}(i_2, v_2)$	$(i_1 < i_2 \leq \lvert s_3 \rvert - 2 \wedge s_3[i_2 + 1] = v_2) \vee$ $(i_1 = i_2 \wedge v_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_3[i_1] = v_1)$	$(i_1 < i_2 \leq s_3.\texttt{size}() - 2 \wedge s_3.\texttt{get}(i_2 + 1) = v_2) \vee$ $(i_1 = i_2 \wedge v_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_3.\texttt{get}(i_1) = v_1)$
	$r_2 = s_2.\texttt{get}(i_2)$	$(i_1 < i_2 < \lvert s_3 \rvert - 1 \wedge r_2 = s_3[i_2 + 1]) \vee$ $(i_1 = i_2 < \lvert s_3 \rvert - 1 \wedge s_3[i_1 + 1] = v_1) \vee$ $i_1 > i_2$	$(i_1 < i_2 < s_3.\texttt{size}() - 1 \wedge r_2 = s_3.\texttt{get}(i_2 + 1)) \vee$ $(i_1 = i_2 < s_3.\texttt{size}() - 1 \wedge s_3.\texttt{get}(i_1 + 1) = v_1) \vee$ $i_1 > i_2$
	$r_2 = s_2.\texttt{indexOf}(v_2)$	$r_2 < 0 \vee$ $0 \leq r_2 < i_1 \vee$ $(r_2 = i_1 \wedge s_3[i + 1] = v_2)$	$r_2 < 0 \vee$ $0 \leq r_2 < i_1 \vee$ $(r_2 = i_1 \wedge s_3.\texttt{get}(i_1 + 1) = v_2)$
	$r_2 = s_2.\texttt{lastIndexOf}(v_2)$	$r_2 < 0 \vee$ $0 \leq r_2 < i_1$	$r_2 < 0 \vee$ $0 \leq r_2 < i_1$
	$r_2 = s_2.\texttt{remove\_at}(i_2)$	$(i_1 < i_2 < \lvert s_3 \rvert \wedge r_2 = s_3[i_2]) \vee$ $(\lvert s_3 \rvert - 1 \geq i_1 = i_2 \wedge s_3[i_1] = v_1) \vee$ $(\lvert s_3 \rvert - 1 \geq i_1 > i_2 \wedge s_3[i_1] = v_1)$	$(i_1 < i_2 < s_3.\texttt{size}() \wedge r_2 = s_3.\texttt{get}(i_2)) \vee$ $(s_3.\texttt{size}() - 1 \geq i_1 = i_2 \wedge s_3.\texttt{get}(i_1) = v_1) \vee$ $(s_3.\texttt{size}() - 1 \geq i_1 > i_2 \wedge s_3.\texttt{get}(i_1) = v_1)$

Table 20: After Commutativity Conditions on ArrayList (Cont.)

	s_2.remove_at(i_2)	$(i_1 < i_2 < \|s_3\| \wedge s_2[i_2] = s_3[i_2]) \vee$ $(\|s_3\| - 1 \geq i_1 = i_2 \wedge s_3[i_1] = v_1) \vee$ $(\|s_3\| - 1 \geq i_1 > i_2 \wedge s_3[i_1] = v_1)$	$(i_1 < i_2 < s_3.\texttt{size}() \wedge s_2.\texttt{get}(i_2) = s_3.\texttt{get}(i_2)) \vee$ $(s_3.\texttt{size}() - 1 \geq i_1 = i_2 \wedge s_3.\texttt{get}(i_1) = v_1) \vee$ $(s_3.\texttt{size}() - 1 \geq i_1 > i_2 \wedge s_3.\texttt{get}(i_1) = v_1)$
	$r_2 = s_2$.set(i_2, v_2)	$(i_1 < i_2 < \|s_3\| - 1 \wedge r_2 = s_3[i_2 + 1] = v_2) \vee$ $(i_1 = i_2 < \|s_3\| - 1 \wedge s_3[i_2 + 1] = v_1 = v_2) \vee$ $i_1 > i_2$	$(i_1 < i_2 < s_3.\texttt{size}() - 1 \wedge r_2 = s_3.\texttt{get}(i_2 + 1) = v_2) \vee$ $(i_1 = i_2 < s_3.\texttt{size}() - 1 \wedge s_3.\texttt{get}(i_2 + 1) = v_1 = v_2) \vee$ $i_1 > i_2$
	s_2.set(i_2, v_2)	$(i_1 < i_2 < \|s_3\| - 1 \wedge s_2[i_2] = s_3[i_2 + 1] = v_2) \vee$ $(i_1 = i_2 < \|s_3\| - 1 \wedge s_3[i_2 + 1] = v_1 = v_2) \vee$ $i_1 > i_2$	$(i_1 < i_2 < s_3.\texttt{size}() - 1 \wedge s_2.\texttt{get}(i_2) = s_3.\texttt{get}(i_2 + 1) = v_2) \vee$ $(i_1 = i_2 < s_3.\texttt{size}() - 1 \wedge s_3.\texttt{get}(i_2 + 1) = v_1 = v_2) \vee$ $i_1 > i_2$
	$r_2 = s_2$.size()	$false$	$false$
$r_1 = s_1$.get(i_1)	s_2.add_at(i_2, v_2)	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge r_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_3[i_1] = r_1)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge r_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_3.\texttt{get}(i_1) = r_1)$
	$r_2 = s_2$.get(i_2)	$true$	$true$
	$r_2 = s_2$.indexOf(v_2)	$true$	$true$
	$r_2 = s_2$.lastIndexOf(v_2)	$true$	$true$
	$r_2 = s_2$.remove_at(i_2)	$i_1 < i_2 \vee$ $(\|s_3\| > i_1 = i_2 \wedge r_1 = s_3[i_1]) \vee$ $(\|s_3\| > i_1 > i_2 \wedge r_1 = s_3[i_1])$	$i_1 < i_2 \vee$ $(s_3.\texttt{size}() > i_1 = i_2 \wedge r_1 = s_3.\texttt{get}(i_1)) \vee$ $(s_3.\texttt{size}() > i_1 > i_2 \wedge r_1 = s_3.\texttt{get}(i_1))$
	s_2.remove_at(i_2)	$i_1 < i_2 \vee$ $(\|s_3\| > i_1 = i_2 \wedge r_1 = s_3[i_1]) \vee$ $(\|s_3\| > i_1 > i_2 \wedge r_1 = s_3[i_1])$	$i_1 < i_2 \vee$ $(s_3.\texttt{size}() > i_1 = i_2 \wedge r_1 = s_3.\texttt{get}(i_1)) \vee$ $(s_3.\texttt{size}() > i_1 > i_2 \wedge r_1 = s_3.\texttt{get}(i_1))$
	$r_2 = s_2$.set(i_2, v_2)	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge r_1 = v_2) \vee$ $i_1 > i_2$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge r_1 = v_2) \vee$ $i_1 > i_2$
	s_2.set(i_2, v_2)	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge r_1 = v_2) \vee$ $i_1 > i_2$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge r_1 = v_2) \vee$ $i_1 > i_2$
$r_1 = s_1$.indexOf(v_1)	s_2.add_at(i_2, v_2)	$(r_1 < 0 \wedge v_1 \neq v_2) \vee$ $0 \leq r_1 < i_2 \vee$ $(r_1 = i_2 \wedge v_1 = v_2)$	$(r_1 < 0 \wedge v_1 \neq v_2) \vee$ $0 \leq r_1 < i_2 \vee$ $(r_1 = i_2 \wedge v_1 = v_2)$
	$r_2 = s_2$.get(i_2)	$true$	$true$
	$r_2 = s_2$.indexOf(v_2)	$true$	$true$
	$r_2 = s_2$.lastIndexOf(v_2)	$true$	$true$
	$r_2 = s_2$.remove_at(i_2)	$r_1 < 0 \vee$ $0 \leq r_1 < i_2 \vee$ $(r_1 = i_2 \wedge i_2 < \|s_3\| \wedge s_3[i_2] = v_1)$	$r_1 < 0 \vee$ $0 \leq r_1 < i_2 \vee$ $(r_1 = i_2 \wedge i_2 < s_3.\texttt{size}() \wedge s_3.\texttt{get}(i_2) = v_1)$
	s_2.remove_at(i_2)	$r_1 < 0 \vee$ $0 \leq r_1 < i_2 \vee$ $(r_1 = i_2 \wedge i_2 < \|s_3\| \wedge s_3[i_2] = v_1)$	$r_1 < 0 \vee$ $0 \leq r_1 < i_2 \vee$ $(r_1 = i_2 \wedge i_2 < s_3.\texttt{size}() \wedge s_3.\texttt{get}(i_2) = v_1)$
	$r_2 = s_2$.set(i_2, v_2)	$(r_1 < 0 \wedge v_1 \neq v_2) \vee$ $0 \leq r_1 < i_2 \vee$ $(r_1 = i_2 \wedge v_1 = v_2) \vee$ $(r_1 > i_2 \wedge v_1 \neq v_2)$	$(r_1 < 0 \wedge v_1 \neq v_2) \vee$ $0 \leq r_1 < i_2 \vee$ $(r_1 = i_2 \wedge v_1 = v_2) \vee$ $(r_1 > i_2 \wedge v_1 \neq v_2)$
	s_2.set(i_2, v_2)	$(r_1 < 0 \wedge v_1 \neq v_2) \vee$ $0 \leq r_1 < i_2 \vee$ $(r_1 = i_2 \wedge v_1 = v_2) \vee$ $(r_1 > i_2 \wedge v_1 \neq v_2)$	$(r_1 < 0 \wedge v_1 \neq v_2) \vee$ $0 \leq r_1 < i_2 \vee$ $(r_1 = i_2 \wedge v_1 = v_2) \vee$ $(r_1 > i_2 \wedge v_1 \neq v_2)$
	$r_2 = s_2$.size()	$true$	$true$
$r_1 = s_1$.lastIndexOf(v_1)	s_2.add_at(i_2, v_2)	$(r_1 < 0 \wedge v_1 \neq v_2) \vee$ $(0 \leq r_1 < i_2 \wedge v_1 \neq v_2)$	$(r_1 < 0 \wedge v_1 \neq v_2) \vee$ $(0 \leq r_1 < i_2 \wedge v_1 \neq v_2)$
	$r_2 = s_2$.get(i_2)	$true$	$true$
	$r_2 = s_2$.indexOf(v_2)	$true$	$true$
	$r_2 = s_2$.lastIndexOf(v_2)	$true$	$true$
	$r_2 = s_2$.remove_at(i_2)	$r_1 < 0 \vee$ $0 \leq r_1 < i_2$	$r_1 < 0 \vee$ $0 \leq r_1 < i_2$
	s_2.remove_at(i_2)	$r_1 < 0 \vee$ $0 \leq r_1 < i_2$	$r_1 < 0 \vee$ $0 \leq r_1 < i_2$
	$r_2 = s_2$.set(i_2, v_2)	$(r_1 < 0 \wedge v_1 \neq v_2) \vee$ $(0 \leq r_1 < i_2 \wedge v_1 \neq v_2) \vee$	$(r_1 < 0 \wedge v_1 \neq v_2) \vee$ $(0 \leq r_1 < i_2 \wedge v_1 \neq v_2) \vee$

Table 20: After Commutativity Conditions on ArrayList (Cont.)

		$(r_1 = i_2 \wedge v_1 = v_2) \vee$ $r_1 > i_2$	$(r_1 = i_2 \wedge v_1 = v_2) \vee$ $r_1 > i_2$
	$s_2.\texttt{set}(i_2, v_2)$	$(r_1 < 0 \wedge v_1 \neq v_2) \vee$ $(0 \leq r_1 < i_2 \wedge v_1 \neq v_2) \vee$ $(r_1 = i_2 \wedge v_1 = v_2) \vee$ $r_1 > i_2$	$(r_1 < 0 \wedge v_1 \neq v_2) \vee$ $(0 \leq r_1 < i_2 \wedge v_1 \neq v_2) \vee$ $(r_1 = i_2 \wedge v_1 = v_2) \vee$ $r_1 > i_2$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$
$r_1 = s_1.\texttt{remove\_at}(i_1)$	$s_2.\texttt{add\_at}(i_2, v_2)$	$(i_1 < i_2 \wedge s_3[i_2 - 1] = v_2) \vee$ $(i_1 = i_2 \wedge r_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_3[i_1] = r_1)$	$(i_1 < i_2 \wedge s_3.\texttt{get}(i_2 - 1) = v_2) \vee$ $(i_1 = i_2 \wedge r_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_3.\texttt{get}(i_1) = r_1)$
	$r_2 = s_2.\texttt{get}(i_2)$	$(i_1 < i_2 \wedge s_3[i_2 - 1] = r_2) \vee$ $(i_1 = i_2 \wedge r_1 = r_2) \vee$ $i_1 > i_2$	$(i_1 < i_2 \wedge s_3.\texttt{get}(i_2 - 1) = r_2) \vee$ $(i_1 = i_2 \wedge r_1 = r_2) \vee$ $i_1 > i_2$
	$r_2 = s_2.\texttt{indexOf}(v_2)$	$(r_2 < 0 \wedge r_1 \neq v_2) \vee$ $0 \leq r_2 < i_1 \vee$ $(r_2 = i_1 \wedge r_1 = v_2 \wedge i_1 < \lvert s_3 \rvert)$	$(r_2 < 0 \wedge r_1 \neq v_2) \vee$ $0 \leq r_2 < i_1 \vee$ $(r_2 = i_1 \wedge r_1 = v_2 \wedge i_1 < s_3.\texttt{size}())$
	$r_2 = s_2.\texttt{lastIndexOf}(v_2)$	$(r_2 < 0 \wedge r_1 \neq v_2) \vee$ $(0 \leq r_2 < i_1 \wedge r_1 \neq v_2)$	$(r_2 < 0 \wedge r_1 \neq v_2) \vee$ $(0 \leq r_2 < i_1 \wedge r_1 \neq v_2)$
	$r_2 = s_2.\texttt{remove\_at}(i_2)$	$(i_1 < i_2 \wedge s_3[i_2 - 1] = r_2) \vee$ $(i_1 = i_2 \wedge r_1 = r_2) \vee$ $(\lvert s_3 \rvert + 1 > i_1 > i_2 \wedge r_1 = s_3[i_1 - 1])$	$(i_1 < i_2 \wedge s_3.\texttt{get}(i_2 - 1) = r_2) \vee$ $(i_1 = i_2 \wedge r_1 = r_2) \vee$ $(s_3.\texttt{size}() + 1 > i_1 > i_2 \wedge r_1 = s_3.\texttt{get}(i_1 - 1))$
	$s_2.\texttt{remove\_at}(i_2)$	$(i_1 < i_2 \wedge s_3[i_2 - 1] = s_2[i_2]) \vee$ $(i_1 = i_2 \wedge r_1 = s_2[i_2]) \vee$ $(\lvert s_3 \rvert + 1 > i_1 > i_2 \wedge r_1 = s_3[i_1 - 1])$	$(i_1 < i_2 \wedge s_3.\texttt{get}(i_2 - 1) = s_2.\texttt{get}(i_2)) \vee$ $(i_1 = i_2 \wedge r_1 = s_2.\texttt{get}(i_2)) \vee$ $(s_3.\texttt{size}() + 1 > i_1 > i_2 \wedge r_1 = s_3.\texttt{get}(i_1 - 1))$
	$r_2 = s_2.\texttt{set}(i_2, v_2)$	$(i_1 < i_2 \wedge s_3[i_2 - 1] = r_2 = v_2) \vee$ $(i_1 = i_2 \wedge r_1 = r_2 = v_2) \vee$ $i_1 > i_2$	$(i_1 < i_2 \wedge s_3.\texttt{get}(i_2 - 1) = r_2 = v_2) \vee$ $(i_1 = i_2 \wedge r_1 = r_2 = v_2) \vee$ $i_1 > i_2$
	$s_2.\texttt{set}(i_2, v_2)$	$(i_1 < i_2 \wedge s_3[i_2 - 1] = s_2[i_2] = v_2) \vee$ $(i_1 = i_2 \wedge r_1 = s_2[i_2] = v_2) \vee$ $i_1 > i_2$	$(i_1 < i_2 \wedge s_3.\texttt{get}(i_2 - 1) = s_2.\texttt{get}(i_2) = v_2) \vee$ $(i_1 = i_2 \wedge r_1 = s_2.\texttt{get}(i_2) = v_2) \vee$ $i_1 > i_2$
	$r_2 = s_2.\texttt{size}()$	$false$	$false$
$s_1.\texttt{remove\_at}(i_1)$	$s_2.\texttt{add\_at}(i_2, v_2)$	$(i_1 < i_2 \wedge s_3[i_2 - 1] = v_2) \vee$ $(i_1 = i_2 \wedge s_1[i_1] = v_2) \vee$ $(i_1 > i_2 \wedge s_3[i_1] = s_1[i_1])$	$(i_1 < i_2 \wedge s_3.\texttt{get}(i_2 - 1) = v_2) \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = v_2) \vee$ $(i_1 > i_2 \wedge s_3.\texttt{get}(i_1) = s_1.\texttt{get}(i_1))$
	$r_2 = s_2.\texttt{get}(i_2)$	$(i_1 < i_2 \wedge s_3[i_2 - 1] = r_2) \vee$ $(i_1 = i_2 \wedge s_1[i_1] = r_2) \vee$ $i_1 > i_2$	$(i_1 < i_2 \wedge s_3.\texttt{get}(i_2 - 1) = r_2) \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = r_2) \vee$ $i_1 > i_2$
	$r_2 = s_2.\texttt{indexOf}(v_2)$	$(r_2 < 0 \wedge s_1[i_1] \neq v_2) \vee$ $0 \leq r_2 < i_1 \vee$ $(r_2 = i_1 \wedge s_1[i_1] = v_2 \wedge i_1 < \lvert s_3 \rvert)$	$(r_2 < 0 \wedge s_1.\texttt{get}(i_1) \neq v_2) \vee$ $0 \leq r_2 < i_1 \vee$ $(r_2 = i_1 \wedge s_1.\texttt{get}(i_1) = v_2 \wedge i_1 < s_3.\texttt{size}())$
	$r_2 = s_2.\texttt{lastIndexOf}(v_2)$	$(r_2 < 0 \wedge s_1[i_1] \neq v_2) \vee$ $(0 \leq r_2 < i_1 \wedge s_1[i_1] \neq v_2)$	$(r_2 < 0 \wedge s_1.\texttt{get}(i_1) \neq v_2) \vee$ $(0 \leq r_2 < i_1 \wedge s_1.\texttt{get}(i_1) \neq v_2)$
	$r_2 = s_2.\texttt{remove\_at}(i_2)$	$(i_1 < i_2 \wedge s_3[i_2 - 1] = r_2) \vee$ $(i_1 = i_2 \wedge s_1[i_1] = r_2) \vee$ $(\lvert s_3 \rvert + 1 > i_1 > i_2 \wedge s_1[i_1] = s_3[i_1 - 1])$	$(i_1 < i_2 \wedge s_3.\texttt{get}(i_2 - 1) = r_2) \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = r_2) \vee$ $(s_3.\texttt{size}() + 1 > i_1 > i_2 \wedge s_1.\texttt{get}(i_1) = s_3.\texttt{get}(i_1 - 1))$
	$s_2.\texttt{remove\_at}(i_2)$	$(i_1 < i_2 \wedge s_3[i_2 - 1] = s_2[i_2]) \vee$ $i_1 = i_2 \vee$ $(\lvert s_3 \rvert + 1 > i_1 > i_2 \wedge s_1[i_1] = s_3[i_1 - 1])$	$(i_1 < i_2 \wedge s_3.\texttt{get}(i_2 - 1) = s_2.\texttt{get}(i_2)) \vee$ $i_1 = i_2 \vee$ $(s_3.\texttt{size}() + 1 > i_1 > i_2 \wedge s_1.\texttt{get}(i_1) = s_3.\texttt{get}(i_1 - 1))$
	$r_2 = s_2.\texttt{set}(i_2, v_2)$	$(i_1 < i_2 \wedge s_3[i_2 - 1] = r_2 = v_2) \vee$ $(i_1 = i_2 \wedge s_1[i_1] = r_2 = v_2) \vee$ $i_1 > i_2$	$(i_1 < i_2 \wedge s_3.\texttt{get}(i_2 - 1) = r_2 = v_2) \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = r_2 = v_2) \vee$ $i_1 > i_2$
	$s_2.\texttt{set}(i_2, v_2)$	$(i_1 < i_2 \wedge s_3[i_2 - 1] = s_2[i_2] = v_2) \vee$ $(i_1 = i_2 \wedge s_2[i_2] = v_2) \vee$ $i_1 > i_2$	$(i_1 < i_2 \wedge s_3.\texttt{get}(i_2 - 1) = s_2.\texttt{get}(i_2) = v_2) \vee$ $(i_1 = i_2 \wedge s_2.\texttt{get}(i_2) = v_2) \vee$ $i_1 > i_2$
	$r_2 = s_2.\texttt{size}()$	$false$	$false$
$r_1 = s_1.\texttt{set}(i_1, v_1)$	$s_2.\texttt{add\_at}(i_2, v_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge r_1 = v_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_3[i_1] = r_1 = v_1)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge r_1 = v_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_3.\texttt{get}(i_1) = r_1 = v_1)$
	$r_2 = s_2.\texttt{get}(i_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge r_1 = v_1) \vee$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge r_1 = v_1) \vee$

Table 20: After Commutativity Conditions on ArrayList (Cont.)

		$i_1 > i_2$	$i_1 > i_2$
	$r_2 = s_2.\texttt{indexOf}(v_2)$	$(r_2 < 0 \wedge r_1 \neq v_2) \vee$ $0 \leq r_2 < i_1 \vee$ $(r_2 = i_1 \wedge r_1 = v_2) \vee$ $(r_2 > i_1 \wedge r_1 \neq v_2)$	$(r_2 < 0 \wedge r_1 \neq v_2) \vee$ $0 \leq r_2 < i_1 \vee$ $(r_2 = i_1 \wedge r_1 = v_2) \vee$ $(r_2 > i_1 \wedge r_1 \neq v_2)$
	$r_2 = s_2.\texttt{lastIndexOf}(v_2)$	$(r_2 < 0 \wedge r_1 \neq v_2) \vee$ $(0 \leq r_2 < i_1 \wedge r_1 \neq v_2) \vee$ $(r_2 = i_1 \wedge r_1 = v_2) \vee$ $r_2 > i_1$	$(r_2 < 0 \wedge r_1 \neq v_2) \vee$ $(0 \leq r_2 < i_1 \wedge r_1 \neq v_2) \vee$ $(r_2 = i_1 \wedge r_1 = v_2) \vee$ $r_2 > i_1$
	$r_2 = s_2.\texttt{remove\_at}(i_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 < \lvert s_3 \rvert \wedge r_1 = s_3[i_1] = v_1) \vee$ $(\lvert s_3 \rvert > i_1 > i_2 \wedge r_1 = s_3[i_1] = v_1)$	$i_1 < i_2 \vee$ $(i_1 = i_2 < s_3.\texttt{size}() \wedge r_1 = s_3.\texttt{get}(i_1) = v_1) \vee$ $(s_3.\texttt{size}() > i_1 > i_2 \wedge r_1 = s_3.\texttt{get}(i_1) = v_1)$
	$s_2.\texttt{remove\_at}(i_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 < \lvert s_3 \rvert \wedge r_1 = s_3[i_1] = v_1) \vee$ $(\lvert s_3 \rvert > i_1 > i_2 \wedge r_1 = s_3[i_1] = v_1)$	$i_1 < i_2 \vee$ $(i_1 = i_2 < s_3.\texttt{size}() \wedge r_1 = s_3.\texttt{get}(i_1) = v_1) \vee$ $(s_3.\texttt{size}() > i_1 > i_2 \wedge r_1 = s_3.\texttt{get}(i_1) = v_1)$
	$r_2 = s_2.\texttt{set}(i_2, v_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge r_1 = v_1 = v_2) \vee$ $i_1 > i_2$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge r_1 = v_1 = v_2) \vee$ $i_1 > i_2$
	$s_2.\texttt{set}(i_2, v_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge r_1 = v_1 = v_2) \vee$ $i_1 > i_2$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge r_1 = v_1 = v_2) \vee$ $i_1 > i_2$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$
$s_1.\texttt{set}(i_1, v_1)$	$s_2.\texttt{add\_at}(i_2, v_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1[i_1] = v_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_3[i_1] = s_1[i_1] = v_1)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = v_1 = v_2) \vee$ $(i_1 > i_2 \wedge s_3.\texttt{get}(i_1) = s_1.\texttt{get}(i_1) = v_1)$
	$r_2 = s_2.\texttt{get}(i_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1[i_1] = v_1) \vee$ $i_1 > i_2$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = v_1) \vee$ $i_1 > i_2$
	$r_2 = s_2.\texttt{indexOf}(v_2)$	$(r_2 < 0 \wedge s_1[i_1] \neq v_2) \vee$ $0 \leq r_2 < i_1 \vee$ $(r_2 = i_1 \wedge s_1[i_1] = v_2) \vee$ $(r_2 > i_1 \wedge s_1[i_1] \neq v_2)$	$(r_2 < 0 \wedge s_1.\texttt{get}(i_1) \neq v_2) \vee$ $0 \leq r_2 < i_1 \vee$ $(r_2 = i_1 \wedge s_1.\texttt{get}(i_1) = v_2) \vee$ $(r_2 > i_1 \wedge s_1.\texttt{get}(i_1) \neq v_2)$
	$r_2 = s_2.\texttt{lastIndexOf}(v_2)$	$(r_2 < 0 \wedge s_1[i_1] \neq v_2) \vee$ $(0 \leq r_2 < i_1 \wedge s_1[i_1] \neq v_2) \vee$ $(r_2 = i_1 \wedge s_1[i_1] = v_2) \vee$ $r_2 > i_1$	$(r_2 < 0 \wedge s_1.\texttt{get}(i_1) \neq v_2) \vee$ $(0 \leq r_2 < i_1 \wedge s_1.\texttt{get}(i_1) \neq v_2) \vee$ $(r_2 = i_1 \wedge s_1.\texttt{get}(i_1) = v_2) \vee$ $r_2 > i_1$
	$r_2 = s_2.\texttt{remove\_at}(i_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 < \lvert s_3 \rvert \wedge s_1[i_1] = s_3[i_1] = v_1) \vee$ $(\lvert s_3 \rvert > i_1 > i_2 \wedge s_1[i_1] = s_3[i_1] = v_1)$	$i_1 < i_2 \vee$ $(i_1 = i_2 < s_3.\texttt{size}() \wedge s_1.\texttt{get}(i_1) = s_3.\texttt{get}(i_1) = v_1) \vee$ $(s_3.\texttt{size}() > i_1 > i_2 \wedge s_1.\texttt{get}(i_1) = s_3.\texttt{get}(i_1) = v_1)$
	$s_2.\texttt{remove\_at}(i_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 < \lvert s_3 \rvert \wedge s_3[i_1] = v_1) \vee$ $(\lvert s_3 \rvert > i_1 > i_2 \wedge s_1[i_1] = s_3[i_1] = v_1)$	$i_1 < i_2 \vee$ $(i_1 = i_2 < s_3.\texttt{size}() \wedge s_3.\texttt{get}(i_1) = v_1) \vee$ $(s_3.\texttt{size}() > i_1 > i_2 \wedge s_1.\texttt{get}(i_1) = s_3.\texttt{get}(i_1) = v_1)$
	$r_2 = s_2.\texttt{set}(i_2, v_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1[i_1] = v_1 = v_2) \vee$ $i_1 > i_2$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1.\texttt{get}(i_1) = v_1 = v_2) \vee$ $i_1 > i_2$
	$s_2.\texttt{set}(i_2, v_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge v_1 = v_2) \vee$ $i_1 > i_2$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge v_1 = v_2) \vee$ $i_1 > i_2$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$
$r_1 = s_1.\texttt{size}(v_1)$	$s_2.\texttt{add\_at}(i_2, v_2)$	$false$	$false$
	$r_2 = s_2.\texttt{get}(i_2)$	$true$	$true$
	$r_2 = s_2.\texttt{indexOf}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{lastIndexOf}(v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{remove\_at}(i_2)$	$false$	$false$
	$s_2.\texttt{remove\_at}(i_2)$	$false$	$false$
	$r_2 = s_2.\texttt{set}(i_2, v_2)$	$true$	$true$
	$s_2.\texttt{set}(i_2, v_2)$	$true$	$true$
	$r_2 = s_2.\texttt{size}()$	$true$	$true$

B. Source Code

B.1 Accumulator

B.1.1 Data Structure

Listing 1. Accumulator.java

```
public class Accumulator {
    private int accumulator;
    /*: public specvar contents :: "int";
        vardefs "contents == accumulator"; */

    public Accumulator(int n)
    /*: modifies "contents"
        ensures "contents = n" */
    {
        accumulator = n;
    }

    public void increase(int n)
    /*: modifies "contents"
        ensures "contents = old contents + n" */
    {
        accumulator = accumulator + n;
    }

    public int read()
    /*: ensures "result = contents" */
    {
        return accumulator;
    }
}
```

B.1.2 Commutativity Testing Methods

Listing 2. AccumulatorComm.java

```
class AccumulatorComm {
    static void increase_increase_pre_s_0(Accumulator sa, Accumulator sb, int n1, int
        n2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
                  sa..contents = sb..contents"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "True" */
        sa.increase(n1);
        sa.increase(n2);

        sb.increase(n2);
        sb.increase(n1);

        /*: assert "sa..contents = sb..contents" */
    }

    static void increase_increase_pre_c_0(Accumulator sa, Accumulator sb, int n1, int
        n2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
                  sa..contents = sb..contents"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "~(True)" */
        sa.increase(n1);
        sa.increase(n2);

```

```
        sb.increase(n2);
        sb.increase(n1);

        /*: assert "~(sa..contents = sb..contents)" */
    }

    static void increase_increase_between_s_1(Accumulator sa, Accumulator sb, int n1,
        int n2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
                  sa..contents = sb..contents"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        sa.increase(n1);
        /*: assume "True" */
        sa.increase(n2);

        sb.increase(n2);
        sb.increase(n1);

        /*: assert "sa..contents = sb..contents" */
    }

    static void increase_increase_between_c_1(Accumulator sa, Accumulator sb, int n1,
        int n2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
                  sa..contents = sb..contents"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        sa.increase(n1);
        /*: assume "~(True)" */
        sa.increase(n2);

        sb.increase(n2);
        sb.increase(n1);

        /*: assert "~(sa..contents = sb..contents)" */
    }

    static void increase_increase_post_s_2(Accumulator sa, Accumulator sb, int n1, int
        n2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
                  sa..contents = sb..contents"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        sa.increase(n1);
        sa.increase(n2);
        /*: assume "True" */

        sb.increase(n2);
        sb.increase(n1);

        /*: assert "sa..contents = sb..contents" */
    }

    static void increase_increase_post_c_2(Accumulator sa, Accumulator sb, int n1, int
        n2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
                  sa..contents = sb..contents"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        sa.increase(n1);
```

```
89          sa.increase(n2);
90          /*: assume "~(True)" */
91
92          sb.increase(n2);
93          sb.increase(n1);
94
95          /*: assert "~(sa..contents = sb..contents)" */
96      }
97
98      static void increase_read_pre_s_3(Accumulator sa, Accumulator sb, int n1)
99      /*: requires "sa ~= null & sb ~= null & sa ~= sb &
100                    sa..contents = sb..contents"
101         modifies "sa..contents", "sb..contents"
102         ensures "True" */
103     {
104         /*: assume "n1 = 0" */
105         sa.increase(n1);
106         int r2a = sa.read();
107
108         int r2b = sb.read();
109         sb.increase(n1);
110
111         /*: assert "r2a = r2b & sa..contents = sb..contents" */
112     }
113
114     static void increase_read_pre_c_3(Accumulator sa, Accumulator sb, int n1)
115     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
116                   sa..contents = sb..contents"
117         modifies "sa..contents", "sb..contents"
118         ensures "True" */
119     {
120         /*: assume "~(n1 = 0)" */
121         sa.increase(n1);
122         int r2a = sa.read();
123
124         int r2b = sb.read();
125         sb.increase(n1);
126
127         /*: assert "~(r2a = r2b & sa..contents = sb..contents)" */
128     }
129
130     static void increase_read_between_s_4(Accumulator sa, Accumulator sb, int n1)
131     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
132                   sa..contents = sb..contents"
133         modifies "sa..contents", "sb..contents"
134         ensures "True" */
135     {
136         sa.increase(n1);
137         /*: assume "n1 = 0" */
138         int r2a = sa.read();
139
140         int r2b = sb.read();
141         sb.increase(n1);
142
143         /*: assert "r2a = r2b & sa..contents = sb..contents" */
144     }
145
146     static void increase_read_between_c_4(Accumulator sa, Accumulator sb, int n1)
147     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
148                   sa..contents = sb..contents"
149         modifies "sa..contents", "sb..contents"
150         ensures "True" */
151     {
152         sa.increase(n1);
153         /*: assume "~(n1 = 0)" */
```

```
154         int r2a = sa.read();
155
156         int r2b = sb.read();
157         sb.increase(n1);
158
159         /*: assert "~(r2a = r2b & sa..contents = sb..contents)" */
160     }
161
162     static void increase_read_post_s_5(Accumulator sa, Accumulator sb, int n1)
163     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
164                   sa..contents = sb..contents"
165         modifies "sa..contents", "sb..contents"
166         ensures "True" */
167     {
168         sa.increase(n1);
169         int r2a = sa.read();
170         /*: assume "n1 = 0" */
171
172         int r2b = sb.read();
173         sb.increase(n1);
174
175         /*: assert "r2a = r2b & sa..contents = sb..contents" */
176     }
177
178     static void increase_read_post_c_5(Accumulator sa, Accumulator sb, int n1)
179     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
180                   sa..contents = sb..contents"
181         modifies "sa..contents", "sb..contents"
182         ensures "True" */
183     {
184         sa.increase(n1);
185         int r2a = sa.read();
186         /*: assume "~(n1 = 0)" */
187
188         int r2b = sb.read();
189         sb.increase(n1);
190
191         /*: assert "~(r2a = r2b & sa..contents = sb..contents)" */
192     }
193
194     static void read_increase_pre_s_6(Accumulator sa, Accumulator sb, int n2)
195     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
196                   sa..contents = sb..contents"
197         modifies "sa..contents", "sb..contents"
198         ensures "True" */
199     {
200         /*: assume "n2 = 0" */
201         int r1a = sa.read();
202         sa.increase(n2);
203
204         sb.increase(n2);
205         int r1b = sb.read();
206
207         /*: assert "r1a = r1b & sa..contents = sb..contents" */
208     }
209
210     static void read_increase_pre_c_6(Accumulator sa, Accumulator sb, int n2)
211     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
212                   sa..contents = sb..contents"
213         modifies "sa..contents", "sb..contents"
214         ensures "True" */
215     {
216         /*: assume "~(n2 = 0)" */
217         int r1a = sa.read();
218         sa.increase(n2);
```

```
219
220         sb.increase(n2);
221         int r1b = sb.read();
222
223         /*: assert "~(r1a = r1b & sa..contents = sb..contents)" */
224     }
225
226     static void read_increase_between_s_7(Accumulator sa, Accumulator sb, int n2)
227     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
228                   sa..contents = sb..contents"
229         modifies "sa..contents", "sb..contents"
230         ensures "True" */
231     {
232         int r1a = sa.read();
233         /*: assume "n2 = 0" */
234         sa.increase(n2);
235
236         sb.increase(n2);
237         int r1b = sb.read();
238
239         /*: assert "r1a = r1b & sa..contents = sb..contents" */
240     }
241
242     static void read_increase_between_c_7(Accumulator sa, Accumulator sb, int n2)
243     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
244                   sa..contents = sb..contents"
245         modifies "sa..contents", "sb..contents"
246         ensures "True" */
247     {
248         int r1a = sa.read();
249         /*: assume "~(n2 = 0)" */
250         sa.increase(n2);
251
252         sb.increase(n2);
253         int r1b = sb.read();
254
255         /*: assert "~(r1a = r1b & sa..contents = sb..contents)" */
256     }
257
258     static void read_increase_post_s_8(Accumulator sa, Accumulator sb, int n2)
259     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
260                   sa..contents = sb..contents"
261         modifies "sa..contents", "sb..contents"
262         ensures "True" */
263     {
264         int r1a = sa.read();
265         sa.increase(n2);
266         /*: assume "n2 = 0" */
267
268         sb.increase(n2);
269         int r1b = sb.read();
270
271         /*: assert "r1a = r1b & sa..contents = sb..contents" */
272     }
273
274     static void read_increase_post_c_8(Accumulator sa, Accumulator sb, int n2)
275     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
276                   sa..contents = sb..contents"
277         modifies "sa..contents", "sb..contents"
278         ensures "True" */
279     {
280         int r1a = sa.read();
281         sa.increase(n2);
282         /*: assume "~(n2 = 0)" */
283
```

```
284         sb.increase(n2);
285         int r1b = sb.read();
286 
287         /*: assert "~(r1a = r1b & sa..contents = sb..contents)" */
288     }
289 
290     static void read_read_pre_s_9(Accumulator sa, Accumulator sb)
291     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
292                   sa..contents = sb..contents"
293         ensures "True" */
294     {
295         /*: assume "True" */
296         int r1a = sa.read();
297         int r2a = sa.read();
298 
299         int r2b = sb.read();
300         int r1b = sb.read();
301 
302         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents" */
303     }
304 
305     static void read_read_pre_c_9(Accumulator sa, Accumulator sb)
306     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
307                   sa..contents = sb..contents"
308         ensures "True" */
309     {
310         /*: assume "~(True)" */
311         int r1a = sa.read();
312         int r2a = sa.read();
313 
314         int r2b = sb.read();
315         int r1b = sb.read();
316 
317         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents)" */
318     }
319 
320     static void read_read_between_s_10(Accumulator sa, Accumulator sb)
321     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
322                   sa..contents = sb..contents"
323         ensures "True" */
324     {
325         int r1a = sa.read();
326         /*: assume "True" */
327         int r2a = sa.read();
328 
329         int r2b = sb.read();
330         int r1b = sb.read();
331 
332         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents" */
333     }
334 
335     static void read_read_between_c_10(Accumulator sa, Accumulator sb)
336     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
337                   sa..contents = sb..contents"
338         ensures "True" */
339     {
340         int r1a = sa.read();
341         /*: assume "~(True)" */
342         int r2a = sa.read();
343 
344         int r2b = sb.read();
345         int r1b = sb.read();
346 
347         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents)" */
348     }
```

```
349
350     static void read_read_post_s_11(Accumulator sa, Accumulator sb)
351     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
352                    sa..contents = sb..contents"
353         ensures "True" */
354     {
355         int r1a = sa.read();
356         int r2a = sa.read();
357         /*: assume "True" */
358
359         int r2b = sb.read();
360         int r1b = sb.read();
361
362         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents" */
363     }
364
365     static void read_read_post_c_11(Accumulator sa, Accumulator sb)
366     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
367                    sa..contents = sb..contents"
368         ensures "True" */
369     {
370         int r1a = sa.read();
371         int r2a = sa.read();
372         /*: assume "~(True)" */
373
374         int r2b = sb.read();
375         int r1b = sb.read();
376
377         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents)" */
378     }
379
380 }
```

B.1.3 Inverse Testing Methods

Listing 3. AccumulatorInv.java

```
1  class AccumulatorInv {
2      static void increase_0(Accumulator s, int v)
3      /*: requires "s ~= null"
4          modifies "s..contents"
5          ensures "True" */
6      {
7          s.increase(v);
8          s.increase(-v);
9
10         /*: assert "s..contents = s..(old contents)" */
11     }
12
13 }
```

B.2 ListSet

B.2.1 Data Structure

Listing 4. ListSet.java

```
1  public /*: claimedby ListSet */ class Node {
2      public Object value;
3      public Node next;
4      /*: public ghost specvar conts :: "objset" = "{}"
5          invariant ContsDef: "this ~= null --> conts = {value} Un next..conts & value ~:
               next..conts"
6          invariant ContsNull: "null..conts = {}" */
7  }
8
```

```
public class ListSet {
    private Node head;
    private int length;

    /*: public specvar contents :: "obj set"
        vardefs "contents == head..conts"

        public specvar size :: "int"
        vardefs "size == length"
        invariant CardInv: "length = card (contents)"

        private static specvar reachable :: "obj => obj => bool"
        vardefs "reachable == (%x y. (x : Node & y = x..next) | (x : ListSet & y =
            x..head))"
        invariant InjInv: "ALL x1 x2 y. y ~= null & reachable x1 y & reachable x2 y -->
            x1 = x2" */

    public ListSet()
    /*: modifies "contents", "size"
        ensures "contents = {} & size = 0" */
    {
        head = null;
        length = 0;

        {
            /*: pickAny x::obj */
            {
                /*: assuming CardHyp: "x : alloc & x : ListSet" */
                {
                    /*: assuming XIsThisHyp: "x = this" */
                    /*: note SizeZero: "x..length = 0" */
                    /*: note ContentsEmpty: "x..contents = {}" */
                    /*: note XIsThisCard: "x..length = card (x..contents)" from
                        XIsThisHyp, SizeZero, ContentsEmpty */
                }
                {
                    /*: assuming XNotThisHyp: "x ~= this" */
                    /*: note XInOldAlloc: "x : old alloc" */
                    /*: note OldCard: "x..(old ListSet.length) = card (x..(old
                        ListSet.contents))" from CardHyp, XNotThisHyp, XInOldAlloc,
                        CardInv */
                    /*: note XSizeEq: "x..length = x..(old ListSet.length)" */
                    /*: note XContentsUnchanged: "x..contents = x..(old
                        ListSet.contents)" */
                    /*: note XNotThisCard: "x..length = card (x..contents)" from
                        XSizeEq, OldCard, XContentsUnchanged */
                }
                /*: note CardConc: "x..length = card (x..contents)" from XIsThisCard,
                    XNotThisCard */
            }
            /*: note CardPostCond: "x : alloc & x : ListSet --> x..length = card
                (x..contents)" forSuch x */
        }
    }

    public boolean add(Object v)
    /*: requires "v ~= null"
        modifies "contents", "size"
        ensures "(v ~: old contents --> contents = old contents Un {v} & result & size
            = old size + 1) & (v : old contents --> contents = old contents & ~result &
            size = old size)" */
    {
        if (_contains(v)) {
            return false;
        } else {
```

```
            _add(v);
            return true;
        }
    }

    private void _add(Object v)
    /*: requires "v ~= null & (v ~: contents) & theinvs"
        modifies "new..Node.value", "new..Node.next", "new..Node.conts", "head",
            "length", "contents", "size"
        ensures "contents = old contents Un {v} & size = old size + 1 & theinvs" */
    {
        Node n = new Node();
        n.value = v;
        n.next = head;
        /*: "n..Node.conts" := "{v} Un head..Node.conts" */
        head = n;

        length = length + 1;

        /*: note ContentsPost: "contents = old contents Un {v}" */
        {
            /*: pickAny x :: obj */
            {
                /*: assuming CardHyp: "x : alloc & x : ListSet" */
                {
                    /*: note ThisProps: "this : old alloc & this : ListSet" */
                    /*: note OldCard: "old length = card (old contents)" from
                        ThisProps, CardInv */
                    /*: note NewSize: "length = old length + 1" */
                    /*: note NewNotInOld: "v ~: old contents" */
                    /*: note XIsThisCard: "length = card (contents)" from OldCard,
                        NewSize, NewNotInOld, ContentsPost */
                }
                {
                    /*: assuming XNotThisHyp: "x ~= this" */
                    /*: note XInOldAlloc: "x : old alloc" */
                    /*: note OldCard: "x..(old ListSet.length) = card (x..(old
                        ListSet.contents))" from CardHyp, XNotThisHyp, XInOldAlloc,
                        CardInv */
                    /*: note XSizeEq: "x..length = x..(old ListSet.length)" */
                    {
                        /*: localize */
                        /*: note XContentsForw: "ALL y. y : x..contents --> y : x..(old
                            ListSet.contents)" */
                        /*: note XContentsBack: "ALL y . y : x..(old ListSet.contents)
                            --> y : x..contents" */
                        /*: note XContentsUnchanged: "x..contents = x..(old
                            ListSet.contents)" from XContentsForw, XContentsBack */
                    }
                    /*: note XNotThisCard: "x..length = card (x..contents)" from
                        XSizeEq, OldCard, XContentsUnchanged */
                }
                /*: note CardConc: "x..length = card (x..contents)" from XIsThisCard,
                    XNotThisCard */
            }
            /*: note CardPostCond: "x : alloc & x : ListSet --> x..length = card
                (x..contents)" forSuch x */
        }
    }

    public boolean contains(Object v)
    /*: ensures "result = (v : contents)" */
    {
        return _contains(v);
    }
```

```
117
118     private boolean _contains(Object v)
119     /*: requires "theinvs"
120         ensures "result = (v : contents) & theinvs" */
121     {
122         Node curr = head;
123         while /*: invariant "((v : contents) = (v : curr..Node.conts))" */ (curr !=
                null) {
124             if (curr.value == v) {
125                 return true;
126             }
127             curr = curr.next;
128         }
129         return false;
130     }
131
132     public boolean remove(Object v)
133     /*: requires "v ~= null"
134         modifies "contents", "size"
135         ensures "(v : old contents --> contents = old contents - {v} & size = old size
                - 1 & result) & (v ~: old contents --> contents = old contents & size = old
                size & ~result)" */
136     {
137         if (_contains(v)) {
138             _remove(v);
139             return true;
140         } else {
141             return false;
142         }
143     }
144
145     private void _remove(Object v)
146     /*: requires "v ~= null & v : contents & theinvs"
147         modifies "Node.next", "Node.conts", "head", "length", "contents", "size"
148         ensures "contents = old contents - {v} & size = old size - 1 & theinvs" */
149     {
150         if (head.value == v) {
151             Node n = head.next;
152             head.next = null;
153             /*: "head..conts" := "{head..value}" */
154             head = n;
155         } else {
156             Node prev = head;
157             /*: "prev..conts" := "prev..conts - {v}" */
158             Node curr = prev.next;
159             while /*: invariant "prev..conts = prev..(old Node.conts) - {v} &
                    prev..next = curr & prev ~= curr & contents = old contents - {v} & (ALL
                    s. s : old Object.alloc & s : ListSet & s ~= this --> s..contents = old
                    (s..contents)) & v : curr..conts & (ALL n. n ~= null & n : Object.alloc
                    & n : Node & n ~= prev --> n..conts = {n..value} Un n..next..conts &
                    (n..value ~: n..next..conts)) & (ALL n. n..conts = n..(old conts) |
                    n..conts = n..(old conts) - {v})" */ (curr.value != v) {
160                 /*: "curr..conts" := "curr..conts - {v}" */
161                 prev = curr;
162                 curr = curr.next;
163             }
164             Node n = curr.next;
165             prev.next = n;
166             curr.next = null;
167             /*: "curr..Node.conts" := "{curr..Node.value}" */
168         }
169
170         length = length - 1;
171
172         /*: note ContentsPost: "contents = old contents - {v}" */
```

```
        {
            /*: pickAny x::obj */
            {
                /*: assuming CardHyp: "x : alloc & x : ListSet" */
                {
                    /*: note ThisProps: "this : old alloc & this : ListSet" */
                    /*: note OldCard: "old length = card (old contents)" from
                        ThisProps, CardInv */
                    /*: note NewSize: "length = old length - 1" */
                    /*: note NewNotInOld: "v : old contents" */
                    /*: note XIsThisCard: "length = card (contents)" from OldCard,
                        NewSize, NewNotInOld, ContentsPost */
                }
                {
                    /*: assuming XNotThisHyp: "x ~= this" */
                    /*: note XInOldAlloc: "x : old alloc" */
                    /*: note OldCard: "x..(old ListSet.length) = card (x..(old
                        ListSet.contents))" from CardHyp, XNotThisHyp, XInOldAlloc,
                        CardInv */
                    /*: note XSizeEq: "x..length = x..(old ListSet.length)" */
                    {
                        /*: localize */
                        /*: note XContentsForw: "ALL y. y : x..contents --> y : x..(old
                            ListSet.contents)" */
                        /*: note XContentsBack: "ALL y. y : x..(old ListSet.contents)
                            --> y : x..contents" */
                        /*: note XContentsUnchanged: "x..contents = x..(old
                            ListSet.contents)" from XContentsForw, XContentsBack */
                    }
                    /*: note XNotThisCard: "x..length = card (x..contents)" from
                        XSizeEq, OldCard, XContentsUnchanged */
                }
                /*: note CardConc: "x..length = card (x..contents)" from XIsThisCard,
                    XNotThisCard */
            }
            /*: note CardPostCond: "x : alloc & x : ListSet --> x..length = card
                (x..contents)" forSuch x */
        }
    }

    public int size()
    /*: ensures "result = size" */
    {
        return length;
    }
}
```

B.2.2 Commutativity Testing Methods

Listing 5. ListSetComm.java

```
class ListSetComm {
    static void add_add_pre_s_0(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v1 ~= v2 | v1 : sa..contents" */
        boolean r1a = sa.add(v1);
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        boolean r1b = sb.add(v1);

```

```
        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void add_add_pre_c_0(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 ~= v2 | v1 : sa..contents)" */
        boolean r1a = sa.add(v1);
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        boolean r1b = sb.add(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void add_add_between_s_1(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.add(v1);
        /*: assume "v1 ~= v2 | ~r1a" */
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        boolean r1b = sb.add(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void add_add_between_c_1(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.add(v1);
        /*: assume "~(v1 ~= v2 | ~r1a)" */
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        boolean r1b = sb.add(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void add_add_post_s_2(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.add(v1);
        boolean r2a = sa.add(v2);
        /*: assume "v1 ~= v2 | ~r1a" */

```

```
76          boolean r2b = sb.add(v2);
77          boolean r1b = sb.add(v1);
78
79          /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
80      }
81
82      static void add_add_post_c_2(ListSet sa, ListSet sb, Object v1, Object v2)
83      /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
84                    sa..contents = sb..contents & sa..size = sb..size"
85          modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
86          ensures "True" */
87      {
88          boolean r1a = sa.add(v1);
89          boolean r2a = sa.add(v2);
90          /*: assume "~(v1 ~= v2 | ~r1a)" */
91
92          boolean r2b = sb.add(v2);
93          boolean r1b = sb.add(v1);
94
95          /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
96      }
97
98      static void add_add_pre_s_3(ListSet sa, ListSet sb, Object v1, Object v2)
99      /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
100                   sa..contents = sb..contents & sa..size = sb..size"
101         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
102         ensures "True" */
103     {
104         /*: assume "v1 ~= v2 | v1 : sa..contents" */
105         boolean r1a = sa.add(v1);
106         sa.add(v2);
107
108         sb.add(v2);
109         boolean r1b = sb.add(v1);
110
111         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
112     }
113
114     static void add_add_pre_c_3(ListSet sa, ListSet sb, Object v1, Object v2)
115     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
116                   sa..contents = sb..contents & sa..size = sb..size"
117         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
118         ensures "True" */
119     {
120         /*: assume "~(v1 ~= v2 | v1 : sa..contents)" */
121         boolean r1a = sa.add(v1);
122         sa.add(v2);
123
124         sb.add(v2);
125         boolean r1b = sb.add(v1);
126
127         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
128     }
129
130     static void add_add_between_s_4(ListSet sa, ListSet sb, Object v1, Object v2)
131     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
132                   sa..contents = sb..contents & sa..size = sb..size"
133         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
134         ensures "True" */
135     {
136         boolean r1a = sa.add(v1);
137         /*: assume "v1 ~= v2 | ~r1a" */
138         sa.add(v2);
```

```
139
140         sb.add(v2);
141         boolean r1b = sb.add(v1);
142
143         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
144     }
145
146     static void add_add_between_c_4(ListSet sa, ListSet sb, Object v1, Object v2)
147     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
148                   sa..contents = sb..contents & sa..size = sb..size"
149         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
150         ensures "True" */
151     {
152         boolean r1a = sa.add(v1);
153         /*: assume "~(v1 ~= v2 | ~r1a)" */
154         sa.add(v2);
155
156         sb.add(v2);
157         boolean r1b = sb.add(v1);
158
159         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
160     }
161
162     static void add_add_post_s_5(ListSet sa, ListSet sb, Object v1, Object v2)
163     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
164                   sa..contents = sb..contents & sa..size = sb..size"
165         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
166         ensures "True" */
167     {
168         boolean r1a = sa.add(v1);
169         sa.add(v2);
170         /*: assume "v1 ~= v2 | ~r1a" */
171
172         sb.add(v2);
173         boolean r1b = sb.add(v1);
174
175         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
176     }
177
178     static void add_add_post_c_5(ListSet sa, ListSet sb, Object v1, Object v2)
179     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
180                   sa..contents = sb..contents & sa..size = sb..size"
181         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
182         ensures "True" */
183     {
184         boolean r1a = sa.add(v1);
185         sa.add(v2);
186         /*: assume "~(v1 ~= v2 | ~r1a)" */
187
188         sb.add(v2);
189         boolean r1b = sb.add(v1);
190
191         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
192     }
193
194     static void add_contains_pre_s_6(ListSet sa, ListSet sb, Object v1, Object v2)
195     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
196                   sa..contents = sb..contents & sa..size = sb..size"
197         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
198         ensures "True" */
199     {
200         /*: assume "v1 ~= v2 | v1 : sa..contents" */
201         boolean r1a = sa.add(v1);
202         boolean r2a = sa.contains(v2);
203
```

```
204         boolean r2b = sb.contains(v2);
205         boolean r1b = sb.add(v1);
206 
207         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
208     }
209 
210     static void add_contains_pre_c_6(ListSet sa, ListSet sb, Object v1, Object v2)
211     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
212                   sa..contents = sb..contents & sa..size = sb..size"
213         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
214         ensures "True" */
215     {
216         /*: assume "~(v1 ~= v2 | v1 : sa..contents)" */
217         boolean r1a = sa.add(v1);
218         boolean r2a = sa.contains(v2);
219 
220         boolean r2b = sb.contains(v2);
221         boolean r1b = sb.add(v1);
222 
223         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
224     }
225 
226     static void add_contains_between_s_7(ListSet sa, ListSet sb, Object v1, Object v2)
227     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
228                   sa..contents = sb..contents & sa..size = sb..size"
229         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
230         ensures "True" */
231     {
232         boolean r1a = sa.add(v1);
233         /*: assume "v1 ~= v2 | ~r1a" */
234         boolean r2a = sa.contains(v2);
235 
236         boolean r2b = sb.contains(v2);
237         boolean r1b = sb.add(v1);
238 
239         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
240     }
241 
242     static void add_contains_between_c_7(ListSet sa, ListSet sb, Object v1, Object v2)
243     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
244                   sa..contents = sb..contents & sa..size = sb..size"
245         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
246         ensures "True" */
247     {
248         boolean r1a = sa.add(v1);
249         /*: assume "~(v1 ~= v2 | ~r1a)" */
250         boolean r2a = sa.contains(v2);
251 
252         boolean r2b = sb.contains(v2);
253         boolean r1b = sb.add(v1);
254 
255         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
256     }
257 
258     static void add_contains_post_s_8(ListSet sa, ListSet sb, Object v1, Object v2)
259     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
260                   sa..contents = sb..contents & sa..size = sb..size"
261         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
262         ensures "True" */
263     {
264         boolean r1a = sa.add(v1);
```

```
265         boolean r2a = sa.contains(v2);
266         /*: assume "v1 ~= v2 | ~r1a" */
267 
268         boolean r2b = sb.contains(v2);
269         boolean r1b = sb.add(v1);
270 
271         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
272     }
273 
274     static void add_contains_post_c_8(ListSet sa, ListSet sb, Object v1, Object v2)
275     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
276                   sa..contents = sb..contents & sa..size = sb..size"
277         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
278         ensures "True" */
279     {
280         boolean r1a = sa.add(v1);
281         boolean r2a = sa.contains(v2);
282         /*: assume "~(v1 ~= v2 | ~r1a)" */
283 
284         boolean r2b = sb.contains(v2);
285         boolean r1b = sb.add(v1);
286 
287         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
288     }
289 
290     static void add_remove_pre_s_9(ListSet sa, ListSet sb, Object v1, Object v2)
291     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
292                   sa..contents = sb..contents & sa..size = sb..size"
293         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
294         ensures "True" */
295     {
296         /*: assume "v1 ~= v2" */
297         boolean r1a = sa.add(v1);
298         boolean r2a = sa.remove(v2);
299 
300         boolean r2b = sb.remove(v2);
301         boolean r1b = sb.add(v1);
302 
303         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
304     }
305 
306     static void add_remove_pre_c_9(ListSet sa, ListSet sb, Object v1, Object v2)
307     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
308                   sa..contents = sb..contents & sa..size = sb..size"
309         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
310         ensures "True" */
311     {
312         /*: assume "~(v1 ~= v2)" */
313         boolean r1a = sa.add(v1);
314         boolean r2a = sa.remove(v2);
315 
316         boolean r2b = sb.remove(v2);
317         boolean r1b = sb.add(v1);
318 
319         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
320     }
321 
322     static void add_remove_between_s_10(ListSet sa, ListSet sb, Object v1, Object v2)
323     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
324                   sa..contents = sb..contents & sa..size = sb..size"
325         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
```

```
        ensures "True" */
    {
        boolean r1a = sa.add(v1);
        /*: assume "v1 ~= v2" */
        boolean r2a = sa.remove(v2);

        boolean r2b = sb.remove(v2);
        boolean r1b = sb.add(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void add_remove_between_c_10(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.add(v1);
        /*: assume "~(v1 ~= v2)" */
        boolean r2a = sa.remove(v2);

        boolean r2b = sb.remove(v2);
        boolean r1b = sb.add(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void add_remove_post_s_11(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.add(v1);
        boolean r2a = sa.remove(v2);
        /*: assume "v1 ~= v2" */

        boolean r2b = sb.remove(v2);
        boolean r1b = sb.add(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void add_remove_post_c_11(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.add(v1);
        boolean r2a = sa.remove(v2);
        /*: assume "~(v1 ~= v2)" */

        boolean r2b = sb.remove(v2);
        boolean r1b = sb.add(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void add_remove_pre_s_12(ListSet sa, ListSet sb, Object v1, Object v2)
```

```
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v1 ~= v2" */
        boolean r1a = sa.add(v1);
        sa.remove(v2);

        sb.remove(v2);
        boolean r1b = sb.add(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void add_remove_pre_c_12(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 ~= v2)" */
        boolean r1a = sa.add(v1);
        sa.remove(v2);

        sb.remove(v2);
        boolean r1b = sb.add(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void add_remove_between_s_13(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.add(v1);
        /*: assume "v1 ~= v2" */
        sa.remove(v2);

        sb.remove(v2);
        boolean r1b = sb.add(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void add_remove_between_c_13(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.add(v1);
        /*: assume "~(v1 ~= v2)" */
        sa.remove(v2);

        sb.remove(v2);
        boolean r1b = sb.add(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void add_remove_post_s_14(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
```

```
452                 sa..contents = sb..contents & sa..size = sb..size"
453         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
454         ensures "True" */
455     {
456         boolean r1a = sa.add(v1);
457         sa.remove(v2);
458         /*: assume "v1 ~= v2" */
459 
460         sb.remove(v2);
461         boolean r1b = sb.add(v1);
462 
463         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
464     }
465 
466     static void add_remove_post_c_14(ListSet sa, ListSet sb, Object v1, Object v2)
467     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
468                 sa..contents = sb..contents & sa..size = sb..size"
469         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
470         ensures "True" */
471     {
472         boolean r1a = sa.add(v1);
473         sa.remove(v2);
474         /*: assume "~(v1 ~= v2)" */
475 
476         sb.remove(v2);
477         boolean r1b = sb.add(v1);
478 
479         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
480     }
481 
482     static void add_size_pre_s_15(ListSet sa, ListSet sb, Object v1)
483     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
484                 sa..contents = sb..contents & sa..size = sb..size"
485         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
486         ensures "True" */
487     {
488         /*: assume "v1 : sa..contents" */
489         boolean r1a = sa.add(v1);
490         int r2a = sa.size();
491 
492         int r2b = sb.size();
493         boolean r1b = sb.add(v1);
494 
495         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
496     }
497 
498     static void add_size_pre_c_15(ListSet sa, ListSet sb, Object v1)
499     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
500                 sa..contents = sb..contents & sa..size = sb..size"
501         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
502         ensures "True" */
503     {
504         /*: assume "~(v1 : sa..contents)" */
505         boolean r1a = sa.add(v1);
506         int r2a = sa.size();
507 
508         int r2b = sb.size();
509         boolean r1b = sb.add(v1);
510 
511         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
512     }
513 
514     static void add_size_between_s_16(ListSet sa, ListSet sb, Object v1)
```

```
515     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
516                 sa..contents = sb..contents & sa..size = sb..size"
517         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
518         ensures "True" */
519     {
520         boolean r1a = sa.add(v1);
521         /*: assume "~r1a" */
522         int r2a = sa.size();
523 
524         int r2b = sb.size();
525         boolean r1b = sb.add(v1);
526 
527         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
528     }
529 
530     static void add_size_between_c_16(ListSet sa, ListSet sb, Object v1)
531     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
532                 sa..contents = sb..contents & sa..size = sb..size"
533         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
534         ensures "True" */
535     {
536         boolean r1a = sa.add(v1);
537         /*: assume "~(~r1a)" */
538         int r2a = sa.size();
539 
540         int r2b = sb.size();
541         boolean r1b = sb.add(v1);
542 
543         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
544     }
545 
546     static void add_size_post_s_17(ListSet sa, ListSet sb, Object v1)
547     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
548                 sa..contents = sb..contents & sa..size = sb..size"
549         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
550         ensures "True" */
551     {
552         boolean r1a = sa.add(v1);
553         int r2a = sa.size();
554         /*: assume "~r1a" */
555 
556         int r2b = sb.size();
557         boolean r1b = sb.add(v1);
558 
559         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
560     }
561 
562     static void add_size_post_c_17(ListSet sa, ListSet sb, Object v1)
563     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
564                 sa..contents = sb..contents & sa..size = sb..size"
565         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
566         ensures "True" */
567     {
568         boolean r1a = sa.add(v1);
569         int r2a = sa.size();
570         /*: assume "~(~r1a)" */
571 
572         int r2b = sb.size();
573         boolean r1b = sb.add(v1);
574 
575         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
```

```
576     }
577
578     static void add_add_pre_s_18(ListSet sa, ListSet sb, Object v1, Object v2)
579     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
580                 sa..contents = sb..contents & sa..size = sb..size"
581         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
582         ensures "True" */
583     {
584         /*: assume "v1 ~= v2 | v1 : sa..contents" */
585         sa.add(v1);
586         boolean r2a = sa.add(v2);
587
588         boolean r2b = sb.add(v2);
589         sb.add(v1);
590
591         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
592     }
593
594     static void add_add_pre_c_18(ListSet sa, ListSet sb, Object v1, Object v2)
595     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
596                 sa..contents = sb..contents & sa..size = sb..size"
597         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
598         ensures "True" */
599     {
600         /*: assume "~(v1 ~= v2 | v1 : sa..contents)" */
601         sa.add(v1);
602         boolean r2a = sa.add(v2);
603
604         boolean r2b = sb.add(v2);
605         sb.add(v1);
606
607         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
608     }
609
610     static void add_add_between_s_19(ListSet sa, ListSet sb, Object v1, Object v2)
611     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
612                 sa..contents = sb..contents & sa..size = sb..size"
613         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
614         ensures "True" */
615     {
616         sa.add(v1);
617         /*: assume "v1 ~= v2 | v1 : sa..(old contents)" */
618         boolean r2a = sa.add(v2);
619
620         boolean r2b = sb.add(v2);
621         sb.add(v1);
622
623         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
624     }
625
626     static void add_add_between_c_19(ListSet sa, ListSet sb, Object v1, Object v2)
627     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
628                 sa..contents = sb..contents & sa..size = sb..size"
629         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
630         ensures "True" */
631     {
632         sa.add(v1);
633         /*: assume "~(v1 ~= v2 | v1 : sa..(old contents))" */
634         boolean r2a = sa.add(v2);
635
636         boolean r2b = sb.add(v2);
637         sb.add(v1);
638
639         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
640     }
```

```
641
642     static void add_add_post_s_20(ListSet sa, ListSet sb, Object v1, Object v2)
643     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
644                  sa..contents = sb..contents & sa..size = sb..size"
645         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
646         ensures "True" */
647     {
648         sa.add(v1);
649         boolean r2a = sa.add(v2);
650         /*: assume "v1 ~= v2 | v1 : sa..(old contents)" */
651
652         boolean r2b = sb.add(v2);
653         sb.add(v1);
654
655         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
656     }
657
658     static void add_add_post_c_20(ListSet sa, ListSet sb, Object v1, Object v2)
659     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
660                  sa..contents = sb..contents & sa..size = sb..size"
661         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
662         ensures "True" */
663     {
664         sa.add(v1);
665         boolean r2a = sa.add(v2);
666         /*: assume "~(v1 ~= v2 | v1 : sa..(old contents))" */
667
668         boolean r2b = sb.add(v2);
669         sb.add(v1);
670
671         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
672     }
673
674     static void add_add_pre_s_21(ListSet sa, ListSet sb, Object v1, Object v2)
675     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
676                  sa..contents = sb..contents & sa..size = sb..size"
677         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
678         ensures "True" */
679     {
680         /*: assume "True" */
681         sa.add(v1);
682         sa.add(v2);
683
684         sb.add(v2);
685         sb.add(v1);
686
687         /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
688     }
689
690     static void add_add_pre_c_21(ListSet sa, ListSet sb, Object v1, Object v2)
691     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
692                  sa..contents = sb..contents & sa..size = sb..size"
693         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
694         ensures "True" */
695     {
696         /*: assume "~(True)" */
697         sa.add(v1);
698         sa.add(v2);
699
700         sb.add(v2);
701         sb.add(v1);
702
703         /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
704     }
705
```

```
706     static void add_add_between_s_22(ListSet sa, ListSet sb, Object v1, Object v2)
707     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
708                   sa..contents = sb..contents & sa..size = sb..size"
709         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
710         ensures "True" */
711     {
712         sa.add(v1);
713         /*: assume "True" */
714         sa.add(v2);
715 
716         sb.add(v2);
717         sb.add(v1);
718 
719         /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
720     }
721 
722     static void add_add_between_c_22(ListSet sa, ListSet sb, Object v1, Object v2)
723     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
724                   sa..contents = sb..contents & sa..size = sb..size"
725         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
726         ensures "True" */
727     {
728         sa.add(v1);
729         /*: assume "~(True)" */
730         sa.add(v2);
731 
732         sb.add(v2);
733         sb.add(v1);
734 
735         /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
736     }
737 
738     static void add_add_post_s_23(ListSet sa, ListSet sb, Object v1, Object v2)
739     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
740                   sa..contents = sb..contents & sa..size = sb..size"
741         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
742         ensures "True" */
743     {
744         sa.add(v1);
745         sa.add(v2);
746         /*: assume "True" */
747 
748         sb.add(v2);
749         sb.add(v1);
750 
751         /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
752     }
753 
754     static void add_add_post_c_23(ListSet sa, ListSet sb, Object v1, Object v2)
755     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
756                   sa..contents = sb..contents & sa..size = sb..size"
757         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
758         ensures "True" */
759     {
760         sa.add(v1);
761         sa.add(v2);
762         /*: assume "~(True)" */
763 
764         sb.add(v2);
765         sb.add(v1);
766 
767         /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
768     }
769 
770     static void add_contains_pre_s_24(ListSet sa, ListSet sb, Object v1, Object v2)
```

```
771     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
772                 sa..contents = sb..contents & sa..size = sb..size"
773         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
774         ensures "True" */
775     {
776         /*: assume "v1 ~= v2 | v1 : sa..contents" */
777         sa.add(v1);
778         boolean r2a = sa.contains(v2);
779 
780         boolean r2b = sb.contains(v2);
781         sb.add(v1);
782 
783         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
784     }
785 
786     static void add_contains_pre_c_24(ListSet sa, ListSet sb, Object v1, Object v2)
787     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
788                 sa..contents = sb..contents & sa..size = sb..size"
789         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
790         ensures "True" */
791     {
792         /*: assume "~(v1 ~= v2 | v1 : sa..contents)" */
793         sa.add(v1);
794         boolean r2a = sa.contains(v2);
795 
796         boolean r2b = sb.contains(v2);
797         sb.add(v1);
798 
799         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
800     }
801 
802     static void add_contains_between_s_25(ListSet sa, ListSet sb, Object v1, Object v2)
803     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
804                 sa..contents = sb..contents & sa..size = sb..size"
805         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
806         ensures "True" */
807     {
808         sa.add(v1);
809         /*: assume "v1 ~= v2 | v1 : sa..(old contents)" */
810         boolean r2a = sa.contains(v2);
811 
812         boolean r2b = sb.contains(v2);
813         sb.add(v1);
814 
815         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
816     }
817 
818     static void add_contains_between_c_25(ListSet sa, ListSet sb, Object v1, Object v2)
819     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
820                 sa..contents = sb..contents & sa..size = sb..size"
821         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
822         ensures "True" */
823     {
824         sa.add(v1);
825         /*: assume "~(v1 ~= v2 | v1 : sa..(old contents))" */
826         boolean r2a = sa.contains(v2);
827 
828         boolean r2b = sb.contains(v2);
829         sb.add(v1);
830 
831         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
832     }
833 
834     static void add_contains_post_s_26(ListSet sa, ListSet sb, Object v1, Object v2)
835     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
```

```
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.add(v1);
        boolean r2a = sa.contains(v2);
        /*: assume "v1 ~= v2 | v1 : sa..(old contents)" */

        boolean r2b = sb.contains(v2);
        sb.add(v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void add_contains_post_c_26(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.add(v1);
        boolean r2a = sa.contains(v2);
        /*: assume "~(v1 ~= v2 | v1 : sa..(old contents))" */

        boolean r2b = sb.contains(v2);
        sb.add(v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void add_remove_pre_s_27(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v1 ~= v2" */
        sa.add(v1);
        boolean r2a = sa.remove(v2);

        boolean r2b = sb.remove(v2);
        sb.add(v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void add_remove_pre_c_27(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 ~= v2)" */
        sa.add(v1);
        boolean r2a = sa.remove(v2);

        boolean r2b = sb.remove(v2);
        sb.add(v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void add_remove_between_s_28(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
```

```
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.add(v1);
        /*: assume "v1 ~= v2" */
        boolean r2a = sa.remove(v2);

        boolean r2b = sb.remove(v2);
        sb.add(v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void add_remove_between_c_28(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.add(v1);
        /*: assume "~(v1 ~= v2)" */
        boolean r2a = sa.remove(v2);

        boolean r2b = sb.remove(v2);
        sb.add(v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void add_remove_post_s_29(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.add(v1);
        boolean r2a = sa.remove(v2);
        /*: assume "v1 ~= v2" */

        boolean r2b = sb.remove(v2);
        sb.add(v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void add_remove_post_c_29(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.add(v1);
        boolean r2a = sa.remove(v2);
        /*: assume "~(v1 ~= v2)" */

        boolean r2b = sb.remove(v2);
        sb.add(v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void add_remove_pre_s_30(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
```

```
        ensures "True" */
    {
        /*: assume "v1 ~= v2" */
        sa.add(v1);
        sa.remove(v2);

        sb.remove(v2);
        sb.add(v1);

        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void add_remove_pre_c_30(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 ~= v2)" */
        sa.add(v1);
        sa.remove(v2);

        sb.remove(v2);
        sb.add(v1);

        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void add_remove_between_s_31(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.add(v1);
        /*: assume "v1 ~= v2" */
        sa.remove(v2);

        sb.remove(v2);
        sb.add(v1);

        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void add_remove_between_c_31(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.add(v1);
        /*: assume "~(v1 ~= v2)" */
        sa.remove(v2);

        sb.remove(v2);
        sb.add(v1);

        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void add_remove_post_s_32(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
```

```
    {
        sa.add(v1);
        sa.remove(v2);
        /*: assume "v1 ~= v2" */

        sb.remove(v2);
        sb.add(v1);

        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void add_remove_post_c_32(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.add(v1);
        sa.remove(v2);
        /*: assume "~(v1 ~= v2)" */

        sb.remove(v2);
        sb.add(v1);

        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void add_size_pre_s_33(ListSet sa, ListSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v1 : sa..contents" */
        sa.add(v1);
        int r2a = sa.size();

        int r2b = sb.size();
        sb.add(v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void add_size_pre_c_33(ListSet sa, ListSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 : sa..contents)" */
        sa.add(v1);
        int r2a = sa.size();

        int r2b = sb.size();
        sb.add(v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void add_size_between_s_34(ListSet sa, ListSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
```

```
        sa.add(v1);
        /*: assume "v1 : sa..(old contents)" */
        int r2a = sa.size();

        int r2b = sb.size();
        sb.add(v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void add_size_between_c_34(ListSet sa, ListSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.add(v1);
        /*: assume "~(v1 : sa..(old contents))" */
        int r2a = sa.size();

        int r2b = sb.size();
        sb.add(v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void add_size_post_s_35(ListSet sa, ListSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.add(v1);
        int r2a = sa.size();
        /*: assume "v1 : sa..(old contents)" */

        int r2b = sb.size();
        sb.add(v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void add_size_post_c_35(ListSet sa, ListSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.add(v1);
        int r2a = sa.size();
        /*: assume "~(v1 : sa..(old contents))" */

        int r2b = sb.size();
        sb.add(v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void contains_add_pre_s_36(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v1 ~= v2 | v1 : sa..contents" */
```

```
        boolean r1a = sa.contains(v1);
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void contains_add_pre_c_36(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 ~= v2 | v1 : sa..contents)" */
        boolean r1a = sa.contains(v1);
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void contains_add_between_s_37(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        /*: assume "v1 ~= v2 | r1a" */
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void contains_add_between_c_37(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        /*: assume "~(v1 ~= v2 | r1a)" */
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void contains_add_post_s_38(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
```

```
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        boolean r2a = sa.add(v2);
        /*: assume "v1 ~= v2 | r1a" */

        boolean r2b = sb.add(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void contains_add_post_c_38(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        boolean r2a = sa.add(v2);
        /*: assume "~(v1 ~= v2 | r1a)" */

        boolean r2b = sb.add(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void contains_add_pre_s_39(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v1 ~= v2 | v1 : sa..contents" */
        boolean r1a = sa.contains(v1);
        sa.add(v2);

        sb.add(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void contains_add_pre_c_39(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 ~= v2 | v1 : sa..contents)" */
        boolean r1a = sa.contains(v1);
        sa.add(v2);

        sb.add(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void contains_add_between_s_40(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
```

```
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        /*: assume "v1 ~= v2 | r1a" */
        sa.add(v2);

        sb.add(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void contains_add_between_c_40(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        /*: assume "~(v1 ~= v2 | r1a)" */
        sa.add(v2);

        sb.add(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void contains_add_post_s_41(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        sa.add(v2);
        /*: assume "v1 ~= v2 | r1a" */

        sb.add(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void contains_add_post_c_41(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        sa.add(v2);
        /*: assume "~(v1 ~= v2 | r1a)" */

        sb.add(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void contains_contains_pre_s_42(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
```

```
    {
        /*: assume "True" */
        boolean r1a = sa.contains(v1);
        boolean r2a = sa.contains(v2);

        boolean r2b = sb.contains(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void contains_contains_pre_c_42(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        /*: assume "~(True)" */
        boolean r1a = sa.contains(v1);
        boolean r2a = sa.contains(v2);

        boolean r2b = sb.contains(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void contains_contains_between_s_43(ListSet sa, ListSet sb, Object v1,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        /*: assume "True" */
        boolean r2a = sa.contains(v2);

        boolean r2b = sb.contains(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void contains_contains_between_c_43(ListSet sa, ListSet sb, Object v1,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        /*: assume "~(True)" */
        boolean r2a = sa.contains(v2);

        boolean r2b = sb.contains(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void contains_contains_post_s_44(ListSet sa, ListSet sb, Object v1, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
```

```
                sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        boolean r2a = sa.contains(v2);
        /*: assume "True" */

        boolean r2b = sb.contains(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void contains_contains_post_c_44(ListSet sa, ListSet sb, Object v1, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
                sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        boolean r2a = sa.contains(v2);
        /*: assume "~(True)" */

        boolean r2b = sb.contains(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void contains_remove_pre_s_45(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v1 ~= v2 | v1 ~: sa..contents" */
        boolean r1a = sa.contains(v1);
        boolean r2a = sa.remove(v2);

        boolean r2b = sb.remove(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void contains_remove_pre_c_45(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 ~= v2 | v1 ~: sa..contents)" */
        boolean r1a = sa.contains(v1);
        boolean r2a = sa.remove(v2);

        boolean r2b = sb.remove(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }
```

```
    static void contains_remove_between_s_46(ListSet sa, ListSet sb, Object v1, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        /*: assume "v1 ~= v2 | ~r1a" */
        boolean r2a = sa.remove(v2);

        boolean r2b = sb.remove(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void contains_remove_between_c_46(ListSet sa, ListSet sb, Object v1, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        /*: assume "~(v1 ~= v2 | ~r1a)" */
        boolean r2a = sa.remove(v2);

        boolean r2b = sb.remove(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void contains_remove_post_s_47(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        boolean r2a = sa.remove(v2);
        /*: assume "v1 ~= v2 | ~r1a" */

        boolean r2b = sb.remove(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void contains_remove_post_c_47(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        boolean r2a = sa.remove(v2);
        /*: assume "~(v1 ~= v2 | ~r1a)" */

        boolean r2b = sb.remove(v2);
        boolean r1b = sb.contains(v1);
```

```
1528
1529        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
1530    }
1531
1532    static void contains_remove_pre_s_48(ListSet sa, ListSet sb, Object v1, Object v2)
1533    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
1534                  sa..contents = sb..contents & sa..size = sb..size"
1535        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1536        ensures "True" */
1537    {
1538        /*: assume "v1 ~= v2 | v1 ~: sa..contents" */
1539        boolean r1a = sa.contains(v1);
1540        sa.remove(v2);
1541
1542        sb.remove(v2);
1543        boolean r1b = sb.contains(v1);
1544
1545        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1546    }
1547
1548    static void contains_remove_pre_c_48(ListSet sa, ListSet sb, Object v1, Object v2)
1549    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
1550                  sa..contents = sb..contents & sa..size = sb..size"
1551        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1552        ensures "True" */
1553    {
1554        /*: assume "~(v1 ~= v2 | v1 ~: sa..contents)" */
1555        boolean r1a = sa.contains(v1);
1556        sa.remove(v2);
1557
1558        sb.remove(v2);
1559        boolean r1b = sb.contains(v1);
1560
1561        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1562    }
1563
1564    static void contains_remove_between_s_49(ListSet sa, ListSet sb, Object v1, Object
        v2)
1565    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
1566                  sa..contents = sb..contents & sa..size = sb..size"
1567        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1568        ensures "True" */
1569    {
1570        boolean r1a = sa.contains(v1);
1571        /*: assume "v1 ~= v2 | ~r1a" */
1572        sa.remove(v2);
1573
1574        sb.remove(v2);
1575        boolean r1b = sb.contains(v1);
1576
1577        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1578    }
1579
1580    static void contains_remove_between_c_49(ListSet sa, ListSet sb, Object v1, Object
        v2)
1581    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
1582                  sa..contents = sb..contents & sa..size = sb..size"
1583        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1584        ensures "True" */
1585    {
1586        boolean r1a = sa.contains(v1);
1587        /*: assume "~(v1 ~= v2 | ~r1a)" */
1588        sa.remove(v2);
1589
```

```
        sb.remove(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void contains_remove_post_s_50(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        sa.remove(v2);
        /*: assume "v1 ~= v2 | ~r1a" */

        sb.remove(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void contains_remove_post_c_50(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        sa.remove(v2);
        /*: assume "~(v1 ~= v2 | ~r1a)" */

        sb.remove(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void contains_size_pre_s_51(ListSet sa, ListSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        /*: assume "True" */
        boolean r1a = sa.contains(v1);
        int r2a = sa.size();

        int r2b = sb.size();
        boolean r1b = sb.contains(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void contains_size_pre_c_51(ListSet sa, ListSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        /*: assume "~(True)" */
        boolean r1a = sa.contains(v1);
        int r2a = sa.size();

        int r2b = sb.size();
        boolean r1b = sb.contains(v1);
```

```
1654
1655        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
1656    }
1657
1658    static void contains_size_between_s_52(ListSet sa, ListSet sb, Object v1)
1659    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
1660                  sa..contents = sb..contents & sa..size = sb..size"
1661        ensures "True" */
1662    {
1663        boolean r1a = sa.contains(v1);
1664        /*: assume "True" */
1665        int r2a = sa.size();
1666
1667        int r2b = sb.size();
1668        boolean r1b = sb.contains(v1);
1669
1670        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
1671    }
1672
1673    static void contains_size_between_c_52(ListSet sa, ListSet sb, Object v1)
1674    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
1675                  sa..contents = sb..contents & sa..size = sb..size"
1676        ensures "True" */
1677    {
1678        boolean r1a = sa.contains(v1);
1679        /*: assume "~(True)" */
1680        int r2a = sa.size();
1681
1682        int r2b = sb.size();
1683        boolean r1b = sb.contains(v1);
1684
1685        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
1686    }
1687
1688    static void contains_size_post_s_53(ListSet sa, ListSet sb, Object v1)
1689    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
1690                  sa..contents = sb..contents & sa..size = sb..size"
1691        ensures "True" */
1692    {
1693        boolean r1a = sa.contains(v1);
1694        int r2a = sa.size();
1695        /*: assume "True" */
1696
1697        int r2b = sb.size();
1698        boolean r1b = sb.contains(v1);
1699
1700        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
1701    }
1702
1703    static void contains_size_post_c_53(ListSet sa, ListSet sb, Object v1)
1704    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
1705                  sa..contents = sb..contents & sa..size = sb..size"
1706        ensures "True" */
1707    {
1708        boolean r1a = sa.contains(v1);
1709        int r2a = sa.size();
1710        /*: assume "~(True)" */
1711
1712        int r2b = sb.size();
1713        boolean r1b = sb.contains(v1);
1714
```

```
        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void remove_add_pre_s_54(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v1 ~= v2" */
        boolean r1a = sa.remove(v1);
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void remove_add_pre_c_54(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 ~= v2)" */
        boolean r1a = sa.remove(v1);
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void remove_add_between_s_55(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        /*: assume "v1 ~= v2" */
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void remove_add_between_c_55(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        /*: assume "~(v1 ~= v2)" */
        boolean r2a = sa.add(v2);
```

```
        boolean r2b = sb.add(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void remove_add_post_s_56(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        boolean r2a = sa.add(v2);
        /*: assume "v1 ~= v2" */

        boolean r2b = sb.add(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void remove_add_post_c_56(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        boolean r2a = sa.add(v2);
        /*: assume "~(v1 ~= v2)" */

        boolean r2b = sb.add(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void remove_add_pre_s_57(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v1 ~= v2" */
        boolean r1a = sa.remove(v1);
        sa.add(v2);

        sb.add(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_add_pre_c_57(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 ~= v2)" */
        boolean r1a = sa.remove(v1);
```

```
        sa.add(v2);

        sb.add(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_add_between_s_58(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        /*: assume "v1 ~= v2" */
        sa.add(v2);

        sb.add(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_add_between_c_58(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        /*: assume "~(v1 ~= v2)" */
        sa.add(v2);

        sb.add(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_add_post_s_59(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        sa.add(v2);
        /*: assume "v1 ~= v2" */

        sb.add(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_add_post_c_59(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        sa.add(v2);
        /*: assume "~(v1 ~= v2)" */
```

```
1903
1904            sb.add(v2);
1905            boolean r1b = sb.remove(v1);
1906
1907            /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1908        }
1909
1910        static void remove_contains_pre_s_60(ListSet sa, ListSet sb, Object v1, Object v2)
1911        /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
1912                     sa..contents = sb..contents & sa..size = sb..size"
1913            modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1914            ensures "True" */
1915        {
1916            /*: assume "v1 ~= v2 | v1 ~: sa..contents" */
1917            boolean r1a = sa.remove(v1);
1918            boolean r2a = sa.contains(v2);
1919
1920            boolean r2b = sb.contains(v2);
1921            boolean r1b = sb.remove(v1);
1922
1923            /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                    sb..size" */
1924        }
1925
1926        static void remove_contains_pre_c_60(ListSet sa, ListSet sb, Object v1, Object v2)
1927        /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
1928                     sa..contents = sb..contents & sa..size = sb..size"
1929            modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1930            ensures "True" */
1931        {
1932            /*: assume "~(v1 ~= v2 | v1 ~: sa..contents)" */
1933            boolean r1a = sa.remove(v1);
1934            boolean r2a = sa.contains(v2);
1935
1936            boolean r2b = sb.contains(v2);
1937            boolean r1b = sb.remove(v1);
1938
1939            /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                    sb..size)" */
1940        }
1941
1942        static void remove_contains_between_s_61(ListSet sa, ListSet sb, Object v1, Object
                v2)
1943        /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
1944                     sa..contents = sb..contents & sa..size = sb..size"
1945            modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1946            ensures "True" */
1947        {
1948            boolean r1a = sa.remove(v1);
1949            /*: assume "v1 ~= v2 | ~r1a" */
1950            boolean r2a = sa.contains(v2);
1951
1952            boolean r2b = sb.contains(v2);
1953            boolean r1b = sb.remove(v1);
1954
1955            /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                    sb..size" */
1956        }
1957
1958        static void remove_contains_between_c_61(ListSet sa, ListSet sb, Object v1, Object
                v2)
1959        /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
1960                     sa..contents = sb..contents & sa..size = sb..size"
1961            modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1962            ensures "True" */
```

```
1963    {
1964        boolean r1a = sa.remove(v1);
1965        /*: assume "~(v1 ~= v2 | ~r1a)" */
1966        boolean r2a = sa.contains(v2);
1967
1968        boolean r2b = sb.contains(v2);
1969        boolean r1b = sb.remove(v1);
1970
1971        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
1972    }
1973
1974    static void remove_contains_post_s_62(ListSet sa, ListSet sb, Object v1, Object v2)
1975    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
1976                  sa..contents = sb..contents & sa..size = sb..size"
1977        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1978        ensures "True" */
1979    {
1980        boolean r1a = sa.remove(v1);
1981        boolean r2a = sa.contains(v2);
1982        /*: assume "v1 ~= v2 | ~r1a" */
1983
1984        boolean r2b = sb.contains(v2);
1985        boolean r1b = sb.remove(v1);
1986
1987        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
1988    }
1989
1990    static void remove_contains_post_c_62(ListSet sa, ListSet sb, Object v1, Object v2)
1991    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
1992                  sa..contents = sb..contents & sa..size = sb..size"
1993        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1994        ensures "True" */
1995    {
1996        boolean r1a = sa.remove(v1);
1997        boolean r2a = sa.contains(v2);
1998        /*: assume "~(v1 ~= v2 | ~r1a)" */
1999
2000        boolean r2b = sb.contains(v2);
2001        boolean r1b = sb.remove(v1);
2002
2003        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
2004    }
2005
2006    static void remove_remove_pre_s_63(ListSet sa, ListSet sb, Object v1, Object v2)
2007    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
2008                  sa..contents = sb..contents & sa..size = sb..size"
2009        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2010        ensures "True" */
2011    {
2012        /*: assume "v1 ~= v2 | v1 ~: sa..contents" */
2013        boolean r1a = sa.remove(v1);
2014        boolean r2a = sa.remove(v2);
2015
2016        boolean r2b = sb.remove(v2);
2017        boolean r1b = sb.remove(v1);
2018
2019        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
2020    }
2021
2022    static void remove_remove_pre_c_63(ListSet sa, ListSet sb, Object v1, Object v2)
2023    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
```

```
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 ~= v2 | v1 ~: sa..contents)" */
        boolean r1a = sa.remove(v1);
        boolean r2a = sa.remove(v2);

        boolean r2b = sb.remove(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void remove_remove_between_s_64(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        /*: assume "v1 ~= v2 | ~r1a" */
        boolean r2a = sa.remove(v2);

        boolean r2b = sb.remove(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void remove_remove_between_c_64(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        /*: assume "~(v1 ~= v2 | ~r1a)" */
        boolean r2a = sa.remove(v2);

        boolean r2b = sb.remove(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void remove_remove_post_s_65(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        boolean r2a = sa.remove(v2);
        /*: assume "v1 ~= v2 | ~r1a" */

        boolean r2b = sb.remove(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }
```

```
2085
2086    static void remove_remove_post_c_65(ListSet sa, ListSet sb, Object v1, Object v2)
2087    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
2088                sa..contents = sb..contents & sa..size = sb..size"
2089        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2090        ensures "True" */
2091    {
2092        boolean r1a = sa.remove(v1);
2093        boolean r2a = sa.remove(v2);
2094        /*: assume "~(v1 ~= v2 | ~r1a)" */
2095
2096        boolean r2b = sb.remove(v2);
2097        boolean r1b = sb.remove(v1);
2098
2099        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
2100    }
2101
2102    static void remove_remove_pre_s_66(ListSet sa, ListSet sb, Object v1, Object v2)
2103    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
2104                sa..contents = sb..contents & sa..size = sb..size"
2105        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2106        ensures "True" */
2107    {
2108        /*: assume "v1 ~= v2 | v1 ~: sa..contents" */
2109        boolean r1a = sa.remove(v1);
2110        sa.remove(v2);
2111
2112        sb.remove(v2);
2113        boolean r1b = sb.remove(v1);
2114
2115        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
2116    }
2117
2118    static void remove_remove_pre_c_66(ListSet sa, ListSet sb, Object v1, Object v2)
2119    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
2120                sa..contents = sb..contents & sa..size = sb..size"
2121        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2122        ensures "True" */
2123    {
2124        /*: assume "~(v1 ~= v2 | v1 ~: sa..contents)" */
2125        boolean r1a = sa.remove(v1);
2126        sa.remove(v2);
2127
2128        sb.remove(v2);
2129        boolean r1b = sb.remove(v1);
2130
2131        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
2132    }
2133
2134    static void remove_remove_between_s_67(ListSet sa, ListSet sb, Object v1, Object v2)
2135    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
2136                sa..contents = sb..contents & sa..size = sb..size"
2137        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2138        ensures "True" */
2139    {
2140        boolean r1a = sa.remove(v1);
2141        /*: assume "v1 ~= v2 | ~r1a" */
2142        sa.remove(v2);
2143
2144        sb.remove(v2);
2145        boolean r1b = sb.remove(v1);
2146
2147        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
2148    }
```

```
2149
2150    static void remove_remove_between_c_67(ListSet sa, ListSet sb, Object v1, Object v2)
2151    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
2152                sa..contents = sb..contents & sa..size = sb..size"
2153        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2154        ensures "True" */
2155    {
2156        boolean r1a = sa.remove(v1);
2157        /*: assume "~(v1 ~= v2 | ~r1a)" */
2158        sa.remove(v2);
2159
2160        sb.remove(v2);
2161        boolean r1b = sb.remove(v1);
2162
2163        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
2164    }
2165
2166    static void remove_remove_post_s_68(ListSet sa, ListSet sb, Object v1, Object v2)
2167    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
2168                sa..contents = sb..contents & sa..size = sb..size"
2169        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2170        ensures "True" */
2171    {
2172        boolean r1a = sa.remove(v1);
2173        sa.remove(v2);
2174        /*: assume "v1 ~= v2 | ~r1a" */
2175
2176        sb.remove(v2);
2177        boolean r1b = sb.remove(v1);
2178
2179        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
2180    }
2181
2182    static void remove_remove_post_c_68(ListSet sa, ListSet sb, Object v1, Object v2)
2183    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
2184                sa..contents = sb..contents & sa..size = sb..size"
2185        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2186        ensures "True" */
2187    {
2188        boolean r1a = sa.remove(v1);
2189        sa.remove(v2);
2190        /*: assume "~(v1 ~= v2 | ~r1a)" */
2191
2192        sb.remove(v2);
2193        boolean r1b = sb.remove(v1);
2194
2195        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
2196    }
2197
2198    static void remove_size_pre_s_69(ListSet sa, ListSet sb, Object v1)
2199    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
2200                sa..contents = sb..contents & sa..size = sb..size"
2201        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2202        ensures "True" */
2203    {
2204        /*: assume "v1 ~: sa..contents" */
2205        boolean r1a = sa.remove(v1);
2206        int r2a = sa.size();
2207
2208        int r2b = sb.size();
2209        boolean r1b = sb.remove(v1);
2210
2211        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
2212    }
```

```
2213
2214    static void remove_size_pre_c_69(ListSet sa, ListSet sb, Object v1)
2215    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
2216                 sa..contents = sb..contents & sa..size = sb..size"
2217        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2218        ensures "True" */
2219    {
2220        /*: assume "~(v1 ~: sa..contents)" */
2221        boolean r1a = sa.remove(v1);
2222        int r2a = sa.size();
2223
2224        int r2b = sb.size();
2225        boolean r1b = sb.remove(v1);
2226
2227        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
2228    }
2229
2230    static void remove_size_between_s_70(ListSet sa, ListSet sb, Object v1)
2231    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
2232                 sa..contents = sb..contents & sa..size = sb..size"
2233        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2234        ensures "True" */
2235    {
2236        boolean r1a = sa.remove(v1);
2237        /*: assume "~r1a" */
2238        int r2a = sa.size();
2239
2240        int r2b = sb.size();
2241        boolean r1b = sb.remove(v1);
2242
2243        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
2244    }
2245
2246    static void remove_size_between_c_70(ListSet sa, ListSet sb, Object v1)
2247    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
2248                 sa..contents = sb..contents & sa..size = sb..size"
2249        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2250        ensures "True" */
2251    {
2252        boolean r1a = sa.remove(v1);
2253        /*: assume "~(~r1a)" */
2254        int r2a = sa.size();
2255
2256        int r2b = sb.size();
2257        boolean r1b = sb.remove(v1);
2258
2259        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
2260    }
2261
2262    static void remove_size_post_s_71(ListSet sa, ListSet sb, Object v1)
2263    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
2264                 sa..contents = sb..contents & sa..size = sb..size"
2265        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2266        ensures "True" */
2267    {
2268        boolean r1a = sa.remove(v1);
2269        int r2a = sa.size();
2270        /*: assume "~r1a" */
2271
2272        int r2b = sb.size();
2273        boolean r1b = sb.remove(v1);
2274
```

```
        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void remove_size_post_c_71(ListSet sa, ListSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        int r2a = sa.size();
        /*: assume "~(~r1a)" */

        int r2b = sb.size();
        boolean r1b = sb.remove(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void remove_add_pre_s_72(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v1 ~= v2" */
        sa.remove(v1);
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        sb.remove(v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_add_pre_c_72(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 ~= v2)" */
        sa.remove(v1);
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        sb.remove(v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_add_between_s_73(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(v1);
        /*: assume "v1 ~= v2" */
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        sb.remove(v1);
```

```
        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_add_between_c_73(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(v1);
        /*: assume "~(v1 ~= v2)" */
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        sb.remove(v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_add_post_s_74(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(v1);
        boolean r2a = sa.add(v2);
        /*: assume "v1 ~= v2" */

        boolean r2b = sb.add(v2);
        sb.remove(v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_add_post_c_74(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(v1);
        boolean r2a = sa.add(v2);
        /*: assume "~(v1 ~= v2)" */

        boolean r2b = sb.add(v2);
        sb.remove(v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_add_pre_s_75(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v1 ~= v2" */
        sa.remove(v1);
        sa.add(v2);

        sb.add(v2);
        sb.remove(v1);
```

```
        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_add_pre_c_75(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 ~= v2)" */
        sa.remove(v1);
        sa.add(v2);

        sb.add(v2);
        sb.remove(v1);

        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_add_between_s_76(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(v1);
        /*: assume "v1 ~= v2" */
        sa.add(v2);

        sb.add(v2);
        sb.remove(v1);

        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_add_between_c_76(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(v1);
        /*: assume "~(v1 ~= v2)" */
        sa.add(v2);

        sb.add(v2);
        sb.remove(v1);

        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_add_post_s_77(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(v1);
        sa.add(v2);
        /*: assume "v1 ~= v2" */

        sb.add(v2);
        sb.remove(v1);

        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
```

```
2468    }
2469
2470    static void remove_add_post_c_77(ListSet sa, ListSet sb, Object v1, Object v2)
2471    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
2472                 sa..contents = sb..contents & sa..size = sb..size"
2473        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2474        ensures "True" */
2475    {
2476        sa.remove(v1);
2477        sa.add(v2);
2478        /*: assume "~(v1 ~= v2)" */
2479
2480        sb.add(v2);
2481        sb.remove(v1);
2482
2483        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
2484    }
2485
2486    static void remove_contains_pre_s_78(ListSet sa, ListSet sb, Object v1, Object v2)
2487    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
2488                 sa..contents = sb..contents & sa..size = sb..size"
2489        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2490        ensures "True" */
2491    {
2492        /*: assume "v1 ~= v2 | v1 ~: sa..contents" */
2493        sa.remove(v1);
2494        boolean r2a = sa.contains(v2);
2495
2496        boolean r2b = sb.contains(v2);
2497        sb.remove(v1);
2498
2499        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2500    }
2501
2502    static void remove_contains_pre_c_78(ListSet sa, ListSet sb, Object v1, Object v2)
2503    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
2504                 sa..contents = sb..contents & sa..size = sb..size"
2505        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2506        ensures "True" */
2507    {
2508        /*: assume "~(v1 ~= v2 | v1 ~: sa..contents)" */
2509        sa.remove(v1);
2510        boolean r2a = sa.contains(v2);
2511
2512        boolean r2b = sb.contains(v2);
2513        sb.remove(v1);
2514
2515        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2516    }
2517
2518    static void remove_contains_between_s_79(ListSet sa, ListSet sb, Object v1, Object
        v2)
2519    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
2520                 sa..contents = sb..contents & sa..size = sb..size"
2521        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2522        ensures "True" */
2523    {
2524        sa.remove(v1);
2525        /*: assume "v1 ~= v2 | v1 ~: sa..(old contents)" */
2526        boolean r2a = sa.contains(v2);
2527
2528        boolean r2b = sb.contains(v2);
2529        sb.remove(v1);
2530
2531        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
```

```
    }

    static void remove_contains_between_c_79(ListSet sa, ListSet sb, Object v1, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(v1);
        /*: assume "~(v1 ~= v2 | v1 ~: sa..(old contents))" */
        boolean r2a = sa.contains(v2);

        boolean r2b = sb.contains(v2);
        sb.remove(v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_contains_post_s_80(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(v1);
        boolean r2a = sa.contains(v2);
        /*: assume "v1 ~= v2 | v1 ~: sa..(old contents)" */

        boolean r2b = sb.contains(v2);
        sb.remove(v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_contains_post_c_80(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(v1);
        boolean r2a = sa.contains(v2);
        /*: assume "~(v1 ~= v2 | v1 ~: sa..(old contents))" */

        boolean r2b = sb.contains(v2);
        sb.remove(v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_remove_pre_s_81(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v1 ~= v2 | v1 ~: sa..contents" */
        sa.remove(v1);
        boolean r2a = sa.remove(v2);

        boolean r2b = sb.remove(v2);
        sb.remove(v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
```

```
    }

    static void remove_remove_pre_c_81(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 ~= v2 | v1 ~: sa..contents)" */
        sa.remove(v1);
        boolean r2a = sa.remove(v2);

        boolean r2b = sb.remove(v2);
        sb.remove(v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_remove_between_s_82(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(v1);
        /*: assume "v1 ~= v2 | v1 ~: sa..(old contents)" */
        boolean r2a = sa.remove(v2);

        boolean r2b = sb.remove(v2);
        sb.remove(v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_remove_between_c_82(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(v1);
        /*: assume "~(v1 ~= v2 | v1 ~: sa..(old contents))" */
        boolean r2a = sa.remove(v2);

        boolean r2b = sb.remove(v2);
        sb.remove(v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_remove_post_s_83(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(v1);
        boolean r2a = sa.remove(v2);
        /*: assume "v1 ~= v2 | v1 ~: sa..(old contents)" */

        boolean r2b = sb.remove(v2);
        sb.remove(v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }
```

```
2661
2662    static void remove_remove_post_c_83(ListSet sa, ListSet sb, Object v1, Object v2)
2663    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
2664                 sa..contents = sb..contents & sa..size = sb..size"
2665        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2666        ensures "True" */
2667    {
2668        sa.remove(v1);
2669        boolean r2a = sa.remove(v2);
2670        /*: assume "~(v1 ~= v2 | v1 ~: sa..(old contents))" */
2671
2672        boolean r2b = sb.remove(v2);
2673        sb.remove(v1);
2674
2675        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2676    }
2677
2678    static void remove_remove_pre_s_84(ListSet sa, ListSet sb, Object v1, Object v2)
2679    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
2680                 sa..contents = sb..contents & sa..size = sb..size"
2681        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2682        ensures "True" */
2683    {
2684        /*: assume "True" */
2685        sa.remove(v1);
2686        sa.remove(v2);
2687
2688        sb.remove(v2);
2689        sb.remove(v1);
2690
2691        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
2692    }
2693
2694    static void remove_remove_pre_c_84(ListSet sa, ListSet sb, Object v1, Object v2)
2695    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
2696                 sa..contents = sb..contents & sa..size = sb..size"
2697        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2698        ensures "True" */
2699    {
2700        /*: assume "~(True)" */
2701        sa.remove(v1);
2702        sa.remove(v2);
2703
2704        sb.remove(v2);
2705        sb.remove(v1);
2706
2707        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
2708    }
2709
2710    static void remove_remove_between_s_85(ListSet sa, ListSet sb, Object v1, Object v2)
2711    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
2712                 sa..contents = sb..contents & sa..size = sb..size"
2713        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2714        ensures "True" */
2715    {
2716        sa.remove(v1);
2717        /*: assume "True" */
2718        sa.remove(v2);
2719
2720        sb.remove(v2);
2721        sb.remove(v1);
2722
2723        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
2724    }
2725
```

```
    static void remove_remove_between_c_85(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(v1);
        /*: assume "~(True)" */
        sa.remove(v2);

        sb.remove(v2);
        sb.remove(v1);

        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_remove_post_s_86(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(v1);
        sa.remove(v2);
        /*: assume "True" */

        sb.remove(v2);
        sb.remove(v1);

        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_remove_post_c_86(ListSet sa, ListSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(v1);
        sa.remove(v2);
        /*: assume "~(True)" */

        sb.remove(v2);
        sb.remove(v1);

        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_size_pre_s_87(ListSet sa, ListSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v1 ~: sa..contents" */
        sa.remove(v1);
        int r2a = sa.size();

        int r2b = sb.size();
        sb.remove(v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_size_pre_c_87(ListSet sa, ListSet sb, Object v1)
```

```
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
                 sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 ~: sa..contents)" */
        sa.remove(v1);
        int r2a = sa.size();

        int r2b = sb.size();
        sb.remove(v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_size_between_s_88(ListSet sa, ListSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
                 sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(v1);
        /*: assume "v1 ~: sa..(old contents)" */
        int r2a = sa.size();

        int r2b = sb.size();
        sb.remove(v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_size_between_c_88(ListSet sa, ListSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
                 sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(v1);
        /*: assume "~(v1 ~: sa..(old contents))" */
        int r2a = sa.size();

        int r2b = sb.size();
        sb.remove(v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_size_post_s_89(ListSet sa, ListSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
                 sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(v1);
        int r2a = sa.size();
        /*: assume "v1 ~: sa..(old contents)" */

        int r2b = sb.size();
        sb.remove(v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_size_post_c_89(ListSet sa, ListSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
```

```
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(v1);
        int r2a = sa.size();
        /*: assume "~(v1 ~: sa..(old contents))" */

        int r2b = sb.size();
        sb.remove(v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void size_add_pre_s_90(ListSet sa, ListSet sb, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v2 : sa..contents" */
        int r1a = sa.size();
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void size_add_pre_c_90(ListSet sa, ListSet sb, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v2 : sa..contents)" */
        int r1a = sa.size();
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void size_add_between_s_91(ListSet sa, ListSet sb, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "v2 : sa..contents" */
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

```

```
2918    static void size_add_between_c_91(ListSet sa, ListSet sb, Object v2)
2919    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
2920                sa..contents = sb..contents & sa..size = sb..size"
2921        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2922        ensures "True" */
2923    {
2924        int r1a = sa.size();
2925        /*: assume "~(v2 : sa..contents)" */
2926        boolean r2a = sa.add(v2);
2927
2928        boolean r2b = sb.add(v2);
2929        int r1b = sb.size();
2930
2931        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
2932    }
2933
2934    static void size_add_post_s_92(ListSet sa, ListSet sb, Object v2)
2935    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
2936                sa..contents = sb..contents & sa..size = sb..size"
2937        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2938        ensures "True" */
2939    {
2940        int r1a = sa.size();
2941        boolean r2a = sa.add(v2);
2942        /*: assume "~r2a" */
2943
2944        boolean r2b = sb.add(v2);
2945        int r1b = sb.size();
2946
2947        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
2948    }
2949
2950    static void size_add_post_c_92(ListSet sa, ListSet sb, Object v2)
2951    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
2952                sa..contents = sb..contents & sa..size = sb..size"
2953        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2954        ensures "True" */
2955    {
2956        int r1a = sa.size();
2957        boolean r2a = sa.add(v2);
2958        /*: assume "~(~r2a)" */
2959
2960        boolean r2b = sb.add(v2);
2961        int r1b = sb.size();
2962
2963        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
2964    }
2965
2966    static void size_add_pre_s_93(ListSet sa, ListSet sb, Object v2)
2967    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
2968                sa..contents = sb..contents & sa..size = sb..size"
2969        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2970        ensures "True" */
2971    {
2972        /*: assume "v2 : sa..contents" */
2973        int r1a = sa.size();
2974        sa.add(v2);
2975
2976        sb.add(v2);
2977        int r1b = sb.size();
2978
2979        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
```

```
    }

    static void size_add_pre_c_93(ListSet sa, ListSet sb, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v2 : sa..contents)" */
        int r1a = sa.size();
        sa.add(v2);

        sb.add(v2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void size_add_between_s_94(ListSet sa, ListSet sb, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "v2 : sa..contents" */
        sa.add(v2);

        sb.add(v2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void size_add_between_c_94(ListSet sa, ListSet sb, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "~(v2 : sa..contents)" */
        sa.add(v2);

        sb.add(v2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void size_add_post_s_95(ListSet sa, ListSet sb, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: ghost specvar sa__contents :: "obj set" = "sa..contents" */
        sa.add(v2);
        /*: assume "v2 : sa__contents" */

        sb.add(v2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
```

```
3045    }
3046
3047    static void size_add_post_c_95(ListSet sa, ListSet sb, Object v2)
3048    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
3049                sa..contents = sb..contents & sa..size = sb..size"
3050        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3051        ensures "True" */
3052    {
3053        int r1a = sa.size();
3054        /*: ghost specvar sa__contents :: "obj set" = "sa..contents" */
3055        sa.add(v2);
3056        /*: assume "~(v2 : sa__contents)" */
3057
3058        sb.add(v2);
3059        int r1b = sb.size();
3060
3061        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
3062    }
3063
3064    static void size_contains_pre_s_96(ListSet sa, ListSet sb, Object v2)
3065    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
3066                sa..contents = sb..contents & sa..size = sb..size"
3067        ensures "True" */
3068    {
3069        /*: assume "True" */
3070        int r1a = sa.size();
3071        boolean r2a = sa.contains(v2);
3072
3073        boolean r2b = sb.contains(v2);
3074        int r1b = sb.size();
3075
3076        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
3077    }
3078
3079    static void size_contains_pre_c_96(ListSet sa, ListSet sb, Object v2)
3080    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
3081                sa..contents = sb..contents & sa..size = sb..size"
3082        ensures "True" */
3083    {
3084        /*: assume "~(True)" */
3085        int r1a = sa.size();
3086        boolean r2a = sa.contains(v2);
3087
3088        boolean r2b = sb.contains(v2);
3089        int r1b = sb.size();
3090
3091        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
3092    }
3093
3094    static void size_contains_between_s_97(ListSet sa, ListSet sb, Object v2)
3095    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
3096                sa..contents = sb..contents & sa..size = sb..size"
3097        ensures "True" */
3098    {
3099        int r1a = sa.size();
3100        /*: assume "True" */
3101        boolean r2a = sa.contains(v2);
3102
3103        boolean r2b = sb.contains(v2);
3104        int r1b = sb.size();
3105
3106        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
```

```
3107    }
3108
3109    static void size_contains_between_c_97(ListSet sa, ListSet sb, Object v2)
3110    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
3111                sa..contents = sb..contents & sa..size = sb..size"
3112        ensures "True" */
3113    {
3114        int r1a = sa.size();
3115        /*: assume "~(True)" */
3116        boolean r2a = sa.contains(v2);
3117
3118        boolean r2b = sb.contains(v2);
3119        int r1b = sb.size();
3120
3121        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
3122    }
3123
3124    static void size_contains_post_s_98(ListSet sa, ListSet sb, Object v2)
3125    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
3126                sa..contents = sb..contents & sa..size = sb..size"
3127        ensures "True" */
3128    {
3129        int r1a = sa.size();
3130        boolean r2a = sa.contains(v2);
3131        /*: assume "True" */
3132
3133        boolean r2b = sb.contains(v2);
3134        int r1b = sb.size();
3135
3136        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
3137    }
3138
3139    static void size_contains_post_c_98(ListSet sa, ListSet sb, Object v2)
3140    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
3141                sa..contents = sb..contents & sa..size = sb..size"
3142        ensures "True" */
3143    {
3144        int r1a = sa.size();
3145        boolean r2a = sa.contains(v2);
3146        /*: assume "~(True)" */
3147
3148        boolean r2b = sb.contains(v2);
3149        int r1b = sb.size();
3150
3151        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
3152    }
3153
3154    static void size_remove_pre_s_99(ListSet sa, ListSet sb, Object v2)
3155    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
3156                sa..contents = sb..contents & sa..size = sb..size"
3157        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3158        ensures "True" */
3159    {
3160        /*: assume "v2 ~: sa..contents" */
3161        int r1a = sa.size();
3162        boolean r2a = sa.remove(v2);
3163
3164        boolean r2b = sb.remove(v2);
3165        int r1b = sb.size();
3166
3167        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
```

```
3168    }
3169
3170    static void size_remove_pre_c_99(ListSet sa, ListSet sb, Object v2)
3171    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
3172                 sa..contents = sb..contents & sa..size = sb..size"
3173        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3174        ensures "True" */
3175    {
3176        /*: assume "~(v2 ~: sa..contents)" */
3177        int r1a = sa.size();
3178        boolean r2a = sa.remove(v2);
3179
3180        boolean r2b = sb.remove(v2);
3181        int r1b = sb.size();
3182
3183        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
3184    }
3185
3186    static void size_remove_between_s_100(ListSet sa, ListSet sb, Object v2)
3187    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
3188                 sa..contents = sb..contents & sa..size = sb..size"
3189        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3190        ensures "True" */
3191    {
3192        int r1a = sa.size();
3193        /*: assume "v2 ~: sa..contents" */
3194        boolean r2a = sa.remove(v2);
3195
3196        boolean r2b = sb.remove(v2);
3197        int r1b = sb.size();
3198
3199        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
3200    }
3201
3202    static void size_remove_between_c_100(ListSet sa, ListSet sb, Object v2)
3203    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
3204                 sa..contents = sb..contents & sa..size = sb..size"
3205        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3206        ensures "True" */
3207    {
3208        int r1a = sa.size();
3209        /*: assume "~(v2 ~: sa..contents)" */
3210        boolean r2a = sa.remove(v2);
3211
3212        boolean r2b = sb.remove(v2);
3213        int r1b = sb.size();
3214
3215        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
3216    }
3217
3218    static void size_remove_post_s_101(ListSet sa, ListSet sb, Object v2)
3219    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
3220                 sa..contents = sb..contents & sa..size = sb..size"
3221        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3222        ensures "True" */
3223    {
3224        int r1a = sa.size();
3225        boolean r2a = sa.remove(v2);
3226        /*: assume "~r2a" */
3227
3228        boolean r2b = sb.remove(v2);
3229        int r1b = sb.size();
```

```
3230
3231        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
3232    }
3233
3234    static void size_remove_post_c_101(ListSet sa, ListSet sb, Object v2)
3235    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
3236                  sa..contents = sb..contents & sa..size = sb..size"
3237        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3238        ensures "True" */
3239    {
3240        int r1a = sa.size();
3241        boolean r2a = sa.remove(v2);
3242        /*: assume "~(~r2a)" */
3243
3244        boolean r2b = sb.remove(v2);
3245        int r1b = sb.size();
3246
3247        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
3248    }
3249
3250    static void size_remove_pre_s_102(ListSet sa, ListSet sb, Object v2)
3251    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
3252                  sa..contents = sb..contents & sa..size = sb..size"
3253        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3254        ensures "True" */
3255    {
3256        /*: assume "v2 ~: sa..contents" */
3257        int r1a = sa.size();
3258        sa.remove(v2);
3259
3260        sb.remove(v2);
3261        int r1b = sb.size();
3262
3263        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
3264    }
3265
3266    static void size_remove_pre_c_102(ListSet sa, ListSet sb, Object v2)
3267    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
3268                  sa..contents = sb..contents & sa..size = sb..size"
3269        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3270        ensures "True" */
3271    {
3272        /*: assume "~(v2 ~: sa..contents)" */
3273        int r1a = sa.size();
3274        sa.remove(v2);
3275
3276        sb.remove(v2);
3277        int r1b = sb.size();
3278
3279        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
3280    }
3281
3282    static void size_remove_between_s_103(ListSet sa, ListSet sb, Object v2)
3283    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
3284                  sa..contents = sb..contents & sa..size = sb..size"
3285        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3286        ensures "True" */
3287    {
3288        int r1a = sa.size();
3289        /*: assume "v2 ~: sa..contents" */
3290        sa.remove(v2);
3291
3292        sb.remove(v2);
```

```
3293        int r1b = sb.size();
3294
3295        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
3296    }
3297
3298    static void size_remove_between_c_103(ListSet sa, ListSet sb, Object v2)
3299    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
3300                sa..contents = sb..contents & sa..size = sb..size"
3301        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3302        ensures "True" */
3303    {
3304        int r1a = sa.size();
3305        /*: assume "~(v2 ~: sa..contents)" */
3306        sa.remove(v2);
3307
3308        sb.remove(v2);
3309        int r1b = sb.size();
3310
3311        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
3312    }
3313
3314    static void size_remove_post_s_104(ListSet sa, ListSet sb, Object v2)
3315    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
3316                sa..contents = sb..contents & sa..size = sb..size"
3317        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3318        ensures "True" */
3319    {
3320        int r1a = sa.size();
3321        /*: ghost specvar sa__contents :: "obj set" = "sa..contents" */
3322        sa.remove(v2);
3323        /*: assume "v2 ~: sa__contents" */
3324
3325        sb.remove(v2);
3326        int r1b = sb.size();
3327
3328        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
3329    }
3330
3331    static void size_remove_post_c_104(ListSet sa, ListSet sb, Object v2)
3332    /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
3333                sa..contents = sb..contents & sa..size = sb..size"
3334        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3335        ensures "True" */
3336    {
3337        int r1a = sa.size();
3338        /*: ghost specvar sa__contents :: "obj set" = "sa..contents" */
3339        sa.remove(v2);
3340        /*: assume "~(v2 ~: sa__contents)" */
3341
3342        sb.remove(v2);
3343        int r1b = sb.size();
3344
3345        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
3346    }
3347
3348    static void size_size_pre_s_105(ListSet sa, ListSet sb)
3349    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
3350                sa..contents = sb..contents & sa..size = sb..size"
3351        ensures "True" */
3352    {
3353        /*: assume "True" */
3354        int r1a = sa.size();
3355        int r2a = sa.size();
3356
3357        int r2b = sb.size();
```

```
3358        int r1b = sb.size();
3359
3360        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
3361    }
3362
3363    static void size_size_pre_c_105(ListSet sa, ListSet sb)
3364    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
3365                  sa..contents = sb..contents & sa..size = sb..size"
3366        ensures "True" */
3367    {
3368        /*: assume "~(True)" */
3369        int r1a = sa.size();
3370        int r2a = sa.size();
3371
3372        int r2b = sb.size();
3373        int r1b = sb.size();
3374
3375        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
3376    }
3377
3378    static void size_size_between_s_106(ListSet sa, ListSet sb)
3379    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
3380                  sa..contents = sb..contents & sa..size = sb..size"
3381        ensures "True" */
3382    {
3383        int r1a = sa.size();
3384        /*: assume "True" */
3385        int r2a = sa.size();
3386
3387        int r2b = sb.size();
3388        int r1b = sb.size();
3389
3390        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
3391    }
3392
3393    static void size_size_between_c_106(ListSet sa, ListSet sb)
3394    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
3395                  sa..contents = sb..contents & sa..size = sb..size"
3396        ensures "True" */
3397    {
3398        int r1a = sa.size();
3399        /*: assume "~(True)" */
3400        int r2a = sa.size();
3401
3402        int r2b = sb.size();
3403        int r1b = sb.size();
3404
3405        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
3406    }
3407
3408    static void size_size_post_s_107(ListSet sa, ListSet sb)
3409    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
3410                  sa..contents = sb..contents & sa..size = sb..size"
3411        ensures "True" */
3412    {
3413        int r1a = sa.size();
3414        int r2a = sa.size();
3415        /*: assume "True" */
3416
3417        int r2b = sb.size();
3418        int r1b = sb.size();
```

```
        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void size_size_post_c_107(ListSet sa, ListSet sb)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
                   sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
        int r2a = sa.size();
        /*: assume "~(True)" */

        int r2b = sb.size();
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

}
```

B.2.3 Inverse Testing Methods

Listing 6. ListSetInv.java

```
class ListSetInv {
    static void add_0(ListSet s, Object v)
    /*: requires "s ~= null & v ~= null"
        modifies "s..contents", "s..size"
        ensures "True" */
    {
        boolean r = s.add(v);
        if (r) { s.remove(v); }

        /*: assert "s..contents = s..(old contents) & s..size = s..(old size)" */
    }

    static void remove_1(ListSet s, Object v)
    /*: requires "s ~= null & v ~= null"
        modifies "s..contents", "s..size"
        ensures "True" */
    {
        boolean r = s.remove(v);
        if (r) { s.add(v); }

        /*: assert "s..contents = s..(old contents) & s..size = s..(old size)" */
    }

}
```

B.3 HashSet

B.3.1 Data Structure

Listing 7. HashSet.java

```
public /*: claimedby HashSet */ class Node {
    public Object value;
    public Node next;

    /*: public ghost specvar conts :: "obj set" = "{}";
        invariant ConDef: "this ~= null --> conts = {value} Un next..conts & (value ~:
            next..conts)";
        invariant ConAlloc: "ALL x. x : conts --> x : alloc";
```

```
        invariant ConNull: "null..conts = {}"; */
}

public class HashSet {
    private Node[] table = null;
    private int _size;

    /*: public ghost specvar init :: "bool" = "False";

        static specvar abs :: "(int => int)"
        vardefs "abs == (%i1. (if (i1 < 0) then (-i1) else i1))";

        static specvar h :: "(obj => int => int)";
        vardefs "h == (%o1. (%i1. ((abs (hashFunc o1)) mod i1)))";

        invariant HashInv: "init --> (ALL k. 0 <= (h k (table..length)) & (h k
            (table..length)) < table..length)";

        public ghost specvar contents :: "obj set" = "{}";
        invariant ContentsDefInv: "init --> contents = {v. v : table.[(h v
            (table..length))]..conts}";

        invariant Coherence: "init --> (ALL i v. 0 <= i & i < table..length --> v :
            table.[i]..conts --> h v (table..length) = i)";

        invariant TableNotNull: "init --> table ~= null";

        invariant TableHidden: "init --> table : hidden";
        invariant NodeHidden1: "init --> (ALL i. 0 <= i & i < table..length & table.[i]
            ~= null --> table.[i] : hidden)";
        invariant NodeHidden2: "ALL n. n : Node & n : alloc & n ~= null & n..next ~=
            null --> n..next : hidden";

        invariant FirstInjInv: "init --> (ALL i x y. y = x..next & y ~= null & 0 <= i &
            i < table..length --> y ~= table.[i])";
        invariant NextInjInv: "ALL x1 x2 y. y ~= null & y = x1..next & y = x2..next -->
            x1 = x2";
        invariant ElementInjInv: "init --> (ALL hs i j. hs : HashSet & hs : alloc &
            hs..init & 0 <= i & i < hs..table..length & 0 <= j & j < table..length &
            hs..table.[i] = table.[j] & table.[j] ~= null --> hs = this & i = j)";
        invariant TableInjInv: "ALL hs. hs..table = table & table ~= null --> hs =
            this";

        public specvar size :: "int"
        vardefs "size == _size"
        invariant CardInv: "init --> _size = card (contents)" */

    public HashSet()
    /*: modifies "contents", "size", "init"
        ensures "init & contents = {} & size = 0" */
    {
        table = new /*: hidden */ Node[11];
        _size = 0;

        /*: "contents" := "{}";
            "init" := "True"; */

        /*: note NewNotHT: "table ~: HashSet"; */
        {
            /*: localize;
                note ElemInj1: "ALL hs1 i j. hs1 : HashSet & hs1 : alloc & hs1..init &
                    0 <= i & i < hs1..table..length & 0 <= j & j < table..length &
                    hs1..table.[i] = table.[j] & hs1..table.[i] ~= null --> hs1 = this &
                    i = j";
```

```
59              note ElemInj2: "ALL hs2 i j. hs2 : HashSet & hs2 : alloc & hs2..init &
                    0 <= i & i < table..length & 0 <= j & j < hs2..table..length &
                    table.[i] = hs2..table.[j] & table.[i] ~= null --> this = hs2 & i =
                    j";
60              note ElemInjOther: "ALL hs1 hs2 i j. hs1 ~= this & hs2 ~= this & hs1 :
                    HashSet & hs1 : alloc & hs1..init & hs2 : HashSet & hs2 : alloc &
                    hs2..init & 0 <= i & i < hs1..table..length & 0 <= j & j <
                    hs2..table..length & hs1..table.[i] = hs2..table.[j] &
                    hs1..table.[i] ~= null --> hs1 = hs2 & i = j" from ElementInjInv,
                    NewNotHT;
61              note ElemInjAll: "theinv ElementInjInv" from ElemInj1, ElemInj2,
                    ElemInjOther; */
62          }
63          {
64              /*: localize;
65              note CohThis: "ALL i v. 0 <= i & i < table..length & v :
                    table.[i]..conts --> h v (table..length) = i"; */ /*: note CohOther:
                    "ALL hs. hs ~= this & hs : alloc & hs : HashSet & hs..init --> (ALL
                    i v. 0 <= i & i < hs..table..length & v : hs..table.[i]..conts --> h
                    v (hs..table..length) = i)" from Coherence, NewNotHT;
66              note CohAll: "theinv Coherence" from CohThis, CohOther; */
67          }
68          {
69              /*: localize;
70              note FirstInjThis: "ALL i x y. y = x..next & y ~= null & 0 <= i & i <
                    table..length --> y ~= table.[i]";
71              note FirstInjOther: "ALL hs. hs : alloc & hs : HashSet & hs ~= this &
                    hs..init --> (ALL i x y. y = x..next & y ~= null & 0 <= i & i <
                    hs..table..length --> y ~= hs..table.[i])" from FirstInjInv,
                    TableInjInv, NewNotHT;
72              note FirstInjAll: "theinv FirstInjInv" from FirstInjThis,
                    FirstInjOther; */
73          }
74          {
75              /*: localize;
76              note TableEmpty: "ALL i. table.[i]..conts = {}";
77              note ContentsThis: "contents = {v. v : table.[(h v
                    (table..length))]..conts}" from TableEmpty;
78              note ContentsOther: "ALL hs. hs : alloc & hs : HashSet & hs ~= this -->
                    hs..contents = old (hs..contents)";
79              note ContentsPost: "theinv ContentsDefInv" from ContentsThis,
                    ContentsOther, ContentsDef, NewNotHT; */
80          }
81          {
82              /*: localize;
83              note NodeHiddenThis: "ALL i. 0 <= i & i < table..length & table.[i] ~=
                    null --> table.[i] : hidden";
84              note NodeHiddenOther: "ALL hs. hs : alloc & hs : HashSet & hs..init &
                    hs ~= this --> (ALL i. 0 <= i & i < hs..table..length &
                    hs..table.[i] ~= null --> hs..table.[i] : hidden)" from NodeHidden1,
                    TableInjInv, NewNotHT;
85              note NodeHiddenAll: "theinv NodeHidden1" from NodeHiddenThis,
                    NodeHiddenOther; */
86          }
87          {
88              /*: localize;
89              note ArrayLength: "0 < table..length";
90              note HashFuncRel: "h = (%o1. (%i1. ((abs (hashFunc o1)) mod i1)))";
91              note HashThis: "ALL v. 0 <= (h v (table..length)) & (h v
                    (table..length)) < table..length" from ArrayLength, HashFuncRel;
92              note HashOther: "ALL hs. hs ~= this & hs : alloc & hs : HashSet &
                    hs..init --> (ALL v. 0 <= (h v (hs..table..length)) & (h v
                    (hs..table..length)) < hs..table..length)" from HashInv, NewNotHT,
                    TableInjInv;
93              note ShowHashInv: "theinv HashInv" from HashThis, HashOther; */
```

```
94          }
95
96          {
97              /*: pickAny x::obj */
98              {
99                  /*: assuming CardHyp: "x : alloc & x : HashSet & x..init" */
100                 {
101                     /*: assuming XIsThisHyp: "x = this" */
102                     /*: note LengthZero: "x.._size = 0" */
103                     /*: note ContentsEmpty: "x..contents = {}" */
104                     /*: note XIsThisCard: "x.._size = card (x..contents)" from
                            XIsThisHyp, LengthZero, ContentsEmpty */
105                 }
106                 {
107                     /*: assuming XNotThisHyp: "x ~= this" */
108                     /*: note XInOldAlloc: "x : old alloc" */
109                     /*: note XOldInit: "x..(old HashSet.init)" */
110                     /*: note OldCard: "x..(old HashSet._size) = card (x..(old
                            HashSet.contents))" from CardHyp, XNotThisHyp, XInOldAlloc,
                            XOldInit, CardInv */
111                     /*: note XLengthEq: "x.._size = x..(old HashSet._size)" */
112                     /*: note XContentsUnchanged: "x..contents = x..(old
                            HashSet.contents)" */
113                     /*: note XNotThisCard: "x.._size = card (x..contents)" from
                            XLengthEq, OldCard, XContentsUnchanged */
114                 }
115                 /*: note CardConc: "x.._size = card (x..contents)" from XIsThisCard,
                        XNotThisCard */
116             }
117             /*: note CardPostCond: "x : alloc & x : HashSet & x..init --> x.._size =
                    card (x..contents)" forSuch x */
118         }
119     }
120
121     private int compute_hash(Object o1)
122     /*: requires "o1 ~= null & init & theinvs"
123         ensures "result = h o1 (table..length) & 0 <= result & result < table..length &
                alloc = old alloc & theinvs" */
124     {
125         int hc = o1.hashCode();
126
127         if (hc < 0) { hc = -hc; }
128
129         /*: note LengthPos: "0 < table..length";
130             note ResLt: "(hc mod table..length) < table..length" from TrueBranch,
                    FalseBranch, LengthPos; */
131
132         return (hc % table.length);
133     }
134
135     public boolean contains(Object v)
136     /*: requires "init & v ~= null"
137         ensures "result = (v : contents)" */
138     {
139         return _contains(v);
140     }
141
142     private boolean _contains(Object v)
143     /*: requires "v ~= null & init & theinvs"
144         ensures "result = (v : contents) & theinvs" */
145     {
146         /*: instantiate ContentsThis: "theinv ContentsDefInv" with "this";
147             mp ContentsRhs: "this : alloc & this : HashSet & init --> contents = {v. v
                    : table.[(h v (table..length))]..conts}"; */
148
```

```
149         int hc = compute_hash(v);
150         boolean res = bucketContains(hc, v);
151 
152         /*: note HC: "hc = h v (table..length)";
153             note InCon: "res = (v : table.[hc]..conts)";
154             note "res = (v : contents)" from InCon, HC, ContentsRhs; */
155 
156         return res;
157     }
158 
159     private boolean bucketContains(int bucket_id, Object v)
160     /*: requires "init & 0 <= bucket_id & bucket_id < table..length & theinvs"
161         ensures "result = (v : table.[bucket_id]..conts) & theinvs" */
162     {
163         Node curr = table[bucket_id];
164         while /*: invariant "(v : table.[bucket_id]..conts) = (v : curr..conts)" */
                (curr != null) {
165             if (curr.value == v)
166                 return true;
167             curr = curr.next;
168         }
169         return false;
170     }
171 
172     public boolean remove(Object v)
173     /*: requires "init & v ~= null"
174         modifies "contents", "size"
175         ensures "(v : old contents --> contents = old contents - {v} & size = old size
                - 1 & result) & (v ~: old contents --> contents = old contents & size = old
                size & ~result)" */
176     {
177         if (_contains(v)) {
178             _remove(v);
179             return true;
180         } else {
181             return false;
182         }
183     }
184 
185     private void removeFirst(Object v, int hc)
186     /*: requires "init & v ~= null & comment ''InContents'' (v : contents) & comment
            ''KFound'' (v = table.[hc]..value) & comment ''HCProps'' (0 <= hc & hc <
            table..length & hc = h v (table..length)) & theinvs"
187         modifies "contents", "size", "conts", "next", "arrayState", "_size"
188         ensures "(contents = old contents - {v}) & (size = old size - 1) & comment
                ''C3'' (ALL a i. a ~= table --> a.[i] = old (a.[i])) & theinvs" */
189     {
190         Node f = table[hc];
191         Node second = f.next;
192 
193         f.next = null;
194         /*: "f..conts" := "{f..value}"; */
195 
196         table[hc] = second;
197         /*: "contents" := "old contents - {v}"; */
198 
199         _size = _size - 1;
200 
201         /*: note ThisProps: "this : alloc & this : HashSet & init";
202             note OldContents: "old contents = {w. w : old (table.[(h w
                    (table..length))]..conts)}" from ContentsDefInv, ThisProps;
203             note FNonNull: "f ~= null";
204             note FProps: "f : Node & f : alloc" from unalloc_lonely, array_pointsto,
                    ThisProps;
```

```
205             note VFound: "v = f..value" from InContents, OldContents, ConDef, KFound,
                    FProps, FNonNull, HCProps;
206             note Acyclic: "fieldRead (old next) f ~= f" from FNonNull, HCProps,
                    FirstInjInv, ThisProps; */
207         {
208             /*: pickAny hs::obj suchThat ContentsDefHyp: "hs : alloc & hs : HashSet &
                    hs..init";
209                 note ContentsThis: "hs = this --> hs..contents = {w. w : hs..table.[(h
                        w (hs..table..length))]..conts}" from OldContents, ElementInjInv,
                        Acyclic, ThisProps, KFound, VFound, ConDef, FProps, FNonNull,
                        HashInv, HCProps; */
210             {
211                 /*: assuming NotThisHyp: "hs ~= this";
212                     note OldHTContents: "fieldRead (old HashSet.contents) hs = {w. w :
                            (fieldRead (old conts) (arrayRead (old arrayState) (hs..table)
                            (h w (hs..table..length))))}" from ContentsDefHyp, NotThisHyp,
                            ContentsDefInv;
213                     note TableNotEq: "hs..table ~= table";
214                     note ContentsOther: "hs..contents = {w.  w : hs..table.[(h w
                            (hs..table..length))]..conts}" from ContentsDefHyp, NotThisHyp,
                            HashInv, ElementInjInv, HCProps, ThisProps, FNonNull,
                            OldHTContents, TableNotEq; */
215             }
216             /*: cases "hs = this", "hs ~= this" for ContentsCases: "hs..contents = {w.
                    w : hs..table.[(h w (hs..table..length))]..conts}" from ContentsThis,
                    ContentsOther;
217                 note ContentsDefPostCond: "hs..contents = {w. w : hs..table.[(h w
                        (hs..table..length))]..conts}" from ContentCases forSuch hs; */
218         }
219         /*: note CoherencePostCond: "theinv Coherence" from Coherence, Acyclic, ConDef,
                ConNull, FNonNull, TableInjInv, FProps, HCProps; */
220
221         /*: note ContentsPost: "contents = old contents - {v}" */
222         {
223             /*: pickAny x::obj */
224             {
225                 /*: assuming CardHyp: "x : alloc & x : HashSet & x..init" */
226                 {
227                     /*: note ThisProps: "this : old alloc & this : HashSet & old init"
                            */
228                     /*: note OldCard: "old _size = card (old contents)" from ThisProps,
                            CardInv */
229                     /*: note NewLength: "_size = old _size - 1" */
230                     /*: note NewNotInOld: "v : old contents" */
231                     /*: note XIsThisCard: "_size = card (contents)" from OldCard,
                            NewLength, NewNotInOld, ContentsPost */
232                 }
233                 {
234                     /*: assuming XNotThisHyp: "x ~= this" */
235                     /*: note XInOldAlloc: "x : old alloc" */
236                     /*: note XOldInit: "x..(old HashSet.init)" */
237                     /*: note OldCard: "x..(old HashSet._size) = card (x..(old
                            HashSet.contents))" from CardHyp, XNotThisHyp, XInOldAlloc,
                            XOldInit, CardInv */
238                     /*: note XLengthEq: "x.._size = x..(old HashSet._size)" */
239                     {
240                         /*: localize */
241                         /*: note XContentsForw: "ALL y. y : x..contents --> y : x..(old
                                HashSet.contents)" */
242                         /*: note XContentsBack: "ALL y. y : x..(old HashSet.contents)
                                --> y : x..contents" */
243                         /*: note XContentsUnchanged: "x..contents = x..(old
                                HashSet.contents)" from XContentsForw, XContentsBack */
244                     }
```

```
245                     /*: note XNotThisCard: "x..|_size = card (x..contents)" from
                            XLengthEq, OldCard, XContentsUnchanged */
246                 }
247                 /*: note CardConc: "x..|_size = card (x..contents)" from XIsThisCard,
                        XNotThisCard */
248             }
249             /*: note CardPostCond: "x : alloc & x : HashSet & x..init --> x..|_size =
                    card (x..contents)" forSuch x */
250         }
251     }
252
253     private Object removeFromBucket(Object v, int hc)
254     /*: requires "comment ''Init'' init & v ~= null & comment ''InContents'' (v :
            contents) & comment ''KNotFound'' (v ~= table.[hc]..value) & comment ''HCProps''
            (0 <= hc & hc < table..length & hc = h v (table..length)) & theinvs"
255         modifies "contents", "size", "conts", "next", "arrayState", "|_size"
256         ensures "(contents = old contents - {v}) & (size = old size - 1) & (ALL a i. a
                ~= table --> a.[i] = old (a.[i])) & theinvs" */
257     {
258         Node f = table[hc];
259         Node prev = f;
260
261         /*: note InBucket: "v : prev..conts" from InContents, ContentsDefInv,
                thisNotNull, thisType, Init, HCProps;
262             note PrevNotNull: "prev ~= null" from InBucket, ConDef, ConNull; */
263
264         /*: "prev..conts" := "prev..conts - {v}"; */
265         /*: "contents" := "old contents - {v}" */
266
267         Node curr = prev.next;
268
269         /*: note PrevHidden: "prev : hidden" from NodeHidden1, thisNotNull, thisType,
                PrevNotNull, Init, HCProps; */
270
271         /*: note ConPreLoop: "ALL n. n : Node & n : alloc & n ~= null & n ~= prev -->
                n..conts = {n..value} Un n..next..conts & (n..value ~: n..next..conts)" from
                ConDef, FirstInjInv, Init, HCProps, thisNotNull, thisType, PrevNotNull;
272             note ConUnchangedPreLoop: "ALL hs i. hs ~= this & hs : HashSet & hs : alloc
                    & hs..init & 0 <= i & i < hs..table..length --> hs..table.[i]..conts =
                    old (hs..table.[i]..conts)" from ElementInjInv, thisType, PrevNotNull,
                    Init, HCProps; */
273         while /*: invariant "prev : Node & prev : alloc & prev ~= null & prev : hidden
                & comment ''PrevCon'' (prev..conts = fieldRead (old conts) prev - {v}) &
                comment ''PrevNot'' (prev..value ~: prev..next..conts) & comment
                ''CurrProps'' (curr : Node & curr : alloc) & comment ''CurrNotNull'' (curr
                ~= null) & comment ''PrevCurr'' (prev..next = curr & prev ~= curr) &
                contents = old contents - {v} & v : curr..conts & comment ''ConDefInv'' (ALL
                n. n : Node & n : alloc & n ~= null & n ~= prev --> n..conts = {n..value} Un
                n..next..conts & n..value ~: n..next..conts) & comment ''ConLoop'' (ALL n.
                n..conts = old (n..conts) | n..conts = old (n..conts) - {v}) & (null..conts
                = {}) & comment ''FConInv'' (f..conts = (fieldRead (old conts) f) - {v}) &
                comment ''ConUnchanged'' (ALL hs i. hs ~= this & hs : HashSet & hs : alloc &
                hs..init & 0 <= i & i < hs..table..length --> hs..table.[i]..conts = old
                (hs..table.[i]..conts))" */ (curr.value != v) {
274             /*: "curr..conts" := "curr..conts - {v}"; */
275
276             /*: note CurrCon: "curr..conts = fieldRead (old conts) curr - {v}";
277                 note PrevIsNot: "prev..value ~= v";
278                 note OldConDef: "fieldRead (old conts) prev = {prev..value} Un
                        fieldRead (old conts) (prev..next)";
279                 note PrevConDef: "prev..conts = {prev..value} Un prev..next..conts"
                        from PrevCurr, PrevCon, CurrCon, OldConDef, PrevIsNot; */
280
281             prev = curr;
282             curr = curr.next;
```

```
283
284             /*: note FConLem: "f..conts = (fieldRead (old conts) f) - {v}" from FConInv;
285
286                 note ConExceptPrev: "ALL n. n : Node & n : alloc & n ~= null & n ~=
                        prev --> n..conts = {n..value} Un n..next..conts & n..value ~:
                        n..next..conts" from PrevConDef, PrevNot, ConDefInv, PrevCurr,
                        NextInjInv, CurrNotNull; */
287         }
288
289         Node tmp = curr.next;
290         prev.next = tmp;
291         curr.next = null;
292
293         /*: "curr..conts" := "{curr..value}"; */
294
295         _size = _size - 1;
296
297         {
298             /*: pickAny x::obj suchThat xHyp: "x : Node & x : alloc & x ~= null"; */
299             {
300                 /*: assuming xIsPrev: "x = prev";
301
302                     note nextNotCurr: "fieldRead (old next) curr ~= curr" from
                            NextInjInv, CurrNotNull, PrevCurr, CurrProps;
303                     note prevNextCon: "prev..next..conts = fieldRead (old conts)
                            (prev..next)";
304                     note prevOldCon: "fieldRead (old conts) prev = {prev..value} Un
                            fieldRead (old conts) curr";
305                     note currOldCon: "fieldRead (old conts) curr = {curr..value} Un
                            fieldRead (old conts) (fieldRead (old next) curr)";
306                     note prevKeyNotK0: "prev..value ~= v"; */
307                 {
308                     /*: pickAny w::obj suchThat ForwHyp: "w : x..conts";
309                         note kNotK0: "w ~= v";
310                         note currKeyIsK0: "curr..value = v";
311                         note ForwCase: "w : {x..value} Un x..next..conts" from xHyp,
                                xIsPrev, ForwHyp, PrevCurr, nextNotCurr, PrevCon,
                                prevNextCon, prevOldCon, currOldCon, prevKeyNotK0, kNotK0,
                                currKeyIsK0 forSuch w; */
312                 }
313                 /*: note BackCase: "ALL w. w : {x..value} Un x..next..conts --> w :
                        x..conts";
314                     note xCon: "x..conts = {x..value} Un x..next..conts" from ForwCase,
                            BackCase; */
315             }
316             /*: cases "x = curr", "x = prev", "x ~= curr & x ~= prev" for XCon:
                    "x..conts = {x..value} Un x..next..conts";
317                 note ConPost: "x..conts = {x..value} Un x..next..conts & x..value ~:
                        x..next..conts" forSuch x; */
318         }
319         {
320             /*: pickAny hs::obj suchThat CohHyp: "hs : alloc & hs : HashSet &
                    hs..init"; */
321             {
322                 /*: pickAny i::int, w::obj suchThat InnerHyp: "0 <= i & i <
                        hs..table..length & w : hs..table.[i]..conts";
323                     note NotCurr: "hs..table.[i] ~= curr";
324                     note InnerConc: "h w (hs..table..length) = i" from CohHyp,
                            InnerHyp, Coherence, NotCurr, ConLoop forSuch i, w; */
325             }
326             /*: note CoherencePost: "(ALL i w. 0 <= i & i < hs..table..length --> w :
                    hs..table.[i]..conts --> h w (hs..table..length) = i)" from InnerConc
                    forSuch hs; */
327         }
328         {
```

```
329             /*: pickAny x::obj suchThat ContentsDefHyp: "x : alloc & x : HashSet &
                    x..init";
330                 note OldXContents: "fieldRead (old HashSet.contents) x = {w. w :
                        fieldRead (old conts) (arrayRead (old arrayState) (x..table) (h w
                        (x..table..length)))}" from ContentsDefHyp, ContentsDefInv; */
331             {
332                 /*: assuming XNotThisHyp: "x ~= this";
333                     note NotCurr: "ALL i. 0 <= i & i < x..table..length -->
                            x..table.[i] ~= curr";
334                     note ConXUnchanged: "ALL i. 0 <= i & i < x..table..length -->
                            x..table.[i]..conts = fieldRead (old conts) (arrayRead (old
                            arrayState) (x..table) i)" from ContentsDefHyp, XNotThisHyp,
                            NotCurr, ConUnchanged;
335                     note LengthLemma: "ALL w. 0 <= (h w (x..table..length)) & (h w
                            (x..table..length)) < (x..table..length)" from ContentsDefHyp,
                            HashInv;
336                     note XNotThisCase: "x..contents = {w. w : x..table.[(h w
                            (x..table..length))]..conts}" from XNotThisHyp, OldXContents,
                            LengthLemma, ConXUnchanged; */
337             }
338             {
339                 /*: assuming XIsThisHyp: "x = this";
340                     note OldContents: "old contents = {w. w : old (table.[(h w
                            (table..length))]..conts)}"; */
341                 {
342                     /*: pickAny w::obj suchThat ForwHyp: "w : contents";
343                         note NotCurr: "table.[(h w (table..length))] ~= curr" from
                                FirstInjInv, ContentsDefHyp,XIsThisHyp, PrevCurr,
                                CurrNotNull, HashInv;
344                         note ForwCase: "w : table.[(h w (table..length))]..conts" from
                                ForwHyp, OldContents, NotCurr, ConLoop forSuch w; */
345                 }
346                 {
347                     /*: pickAny w::obj suchThat BackHyp: "w : table.[(h w
                            (table..length))]..conts";
348                         note NotCurr: "table.[(h w (table..length))] ~= curr" from
                                FirstInjInv, ContentsDefHyp, XIsThisHyp, PrevCurr,
                                CurrNotNull, HashInv;
349                         note BackCase: "w : contents" from BackHyp, OldContents,
                                NotCurr, ConLoop, FConInv, HCProps forSuch w; */
350                 }
351                 /*: note XIsThisCase: "contents = {w. w : table.[(h w
                        (table..length))]..conts}" from ForwCase, BackCase; */
352             }
353             /*: note ContentsDefPost: "x..contents = {w. w : x..table.[(h w
                    (x..table..length))]..conts}" from XNotThisCase, XIsThisCase forSuch x;
                    */
354         }
355
356         /*: note ContentsPost: "contents = old contents - {v}" */
357         {
358             /*: pickAny x::obj */
359             {
360                 /*: assuming CardHyp: "x : alloc & x : HashSet & x..init" */
361                 {
362                     /*: note ThisProps: "this : old alloc & this : HashSet & old init"
                            */
363                     /*: note OldCard: "old _size = card (old contents)" from ThisProps,
                            CardInv */
364                     /*: note NewLength: "_size = old _size - 1" */
365                     /*: note NewNotInOld: "v : old contents" */
366                     /*: note XIsThisCard: "_size = card (contents)" from OldCard,
                            NewLength, NewNotInOld, ContentsPost */
367                 }
368                 {
```

```
                    /*: assuming XNotThisHyp: "x ~= this" */
                    /*: note XInOldAlloc: "x : old alloc" */
                    /*: note XOldInit: "x..(old HashSet.init)" */
                    /*: note OldCard: "x..(old HashSet._size) = card (x..(old
                        HashSet.contents))" from CardHyp, XNotThisHyp, XInOldAlloc,
                        XOldInit, CardInv */
                    /*: note XLengthEq: "x.._size = x..(old HashSet._size)" */
                    {
                        /*: localize */
                        /*: note XContentsForw: "ALL y. y : x..contents --> y : x..(old
                            HashSet.contents)" */
                        /*: note XContentsBack: "ALL y. y : x..(old HashSet.contents)
                            --> y : x..contents" */
                        /*: note XContentsUnchanged: "x..contents = x..(old
                            HashSet.contents)" from XContentsForw, XContentsBack */
                    }
                    /*: note XNotThisCard: "x.._size = card (x..contents)" from
                        XLengthEq, OldCard, XContentsUnchanged */
                }
                /*: note CardConc: "x.._size = card (x..contents)" from XIsThisCard,
                    XNotThisCard */
            }
            /*: note CardPostCond: "x : alloc & x : HashSet & x..init --> x.._size =
                card (x..contents)" forSuch x */
        }
    }

    private void _remove(Object v)
    /*: requires "(comment ''Init'' init) & v ~= null & (comment ''KeyInContents'' (v :
        contents)) & theinvs"
        modifies "contents", "size", "conts", "next", "arrayState", "_size"
        ensures "(contents = old contents - {v}) & (size = old size - 1) & (v : old
            contents) & (ALL a i. a ~= table --> a.[i] = old (a.[i])) & theinvs" */
    {
        int hc = compute_hash(v);
        Node f = table[hc];

        /*: note ThisProps: "this : alloc & this : HashSet";
            note HCDef: "hc = h v (table..length)";
            note KeyInBucket: "v : table.[hc]..conts" from HCDef, KeyInContents,
                ContentsDef, Init, ThisProps; */

        if (f.value == v) {
            removeFirst(v, hc);
        } else {
            removeFromBucket(v, hc);
        }
    }

    private void _add(Object v)
    /*: requires "comment ''Init'' init & v ~= null & (v ~: contents) & theinvs"
        modifies "contents", "size", "arrayState", "new..conts", "new..next",
            "new..value", "_size"
        ensures  "contents = old contents Un {v} & size = old size + 1 & (ALL a i. a ~=
            table --> a.[i] = old (a.[i])) & theinvs" */
    {
        int hc = compute_hash(v);
        Node n = new /*: hidden */ Node();
        n.value = v;
        Node first = table[hc];
        n.next = first;
        /*: "n..conts" := "{v} Un first..conts"; */
        table[hc] = n;
        /*: "contents" := "(old contents) Un {v}"; */

```

```
421         _size = _size + 1;
422
423         /*: note NewNotHT: "n ~: HashSet";
424             note ThisProps: "this : alloc & this : old alloc & this : HashSet";
425             note HCBounds: "0 <= hc & hc < table..length";
426             note NewOldNEq: "n ~= first";
427             note ThisTableNotNull: "table ~= null";
428             note OldNotRefInTable: "ALL hs i. hs : alloc & hs : HashSet & hs..init & 0
                    <= i & i < hs..table..length & first ~= null --> hs..table.[i] ~= first"
                    from OldNotRefInTable, Init, ThisProps, HCBounds, ElementInjInvHash,
                    NewOldNEq, TableInjInvHash, NewNotHT, ThisTableNotNull; */
429
430         /*: note HashPost: "theinv HashInv" from HashPost, HashInv, NewNotHT; */
431
432         /*: note ValAlloc: "v : alloc";
433             note AllocChanged: "alloc = old alloc Un {n}";
434             note FirstProps: "first : alloc & first : Node & first ~= n" from
                    unalloc_lonely, AllocChanged, ThisProps, array_pointsto, NewNotHT; */
435
436         /*: note ConAllocLemma: "theinv ConAlloc" from ConAllocLemma, ConAlloc,
                ValAlloc, FirstProps; */
437
438         /*: note NewHidden2: "n..next ~= null --> n..next : hidden" from NewHidden2,
                NodeHidden1, ThisProps, Init, HCBounds;
439             note OldHidden2: "ALL m. m ~= n & m : Node & m : alloc & m ~= null &
                    m..next ~= null --> m..next : hidden";
440             note AllHidden2: "theinv NodeHidden2" from AllHidden2, NewHidden2,
                    OldHidden2; */
441
442         /*: note NewHidden: "n : hidden";
443             note ThisHidden1: "ALL i. 0 <= i & i < table..length & table.[i] ~= null
                    --> table.[i] : hidden" from ThisHidden1, NodeHidden1, NewHidden,
                    ThisProps, Init;
444             note OtherHidden1: "ALL hs. hs ~= this & hs : alloc & hs : HashSet &
                    hs..init --> (ALL i. 0 <= i & i < hs..table..length & hs..table.[i] ~=
                    null --> hs..table.[i] : hidden)" from OtherHidden1, NodeHidden1,
                    NewNotHT;
445             note Hidden1All: "theinv NodeHidden1" from Hidden1All, ThisHidden1,
                    OtherHidden1; */
446
447         /*: note AllocChange: "Object.alloc = old Object.alloc Un {n}";
448             note HProp: "hc = h v (table..Array.length)"; */
449
450         /*: note NewNotRefThisArr: "ALL i. 0 <= i & i < table..length --> (arrayRead
                (old arrayState) table i) ~= n";*/
451         /*: note NewNotRefArray: "ALL a i. 0 <= i & i < a..length --> (arrayRead (old
                arrayState) a i) ~= n"; */
452
453         /*: note NewNotAlloc: "n ~: old alloc";
454             note NewNotRefByNext: "ALL x. x..next ~= n" from NewOldNEq, NewNotAlloc,
                    unalloc_lonely, NewNotRefByNext; */
455
456         /*: note NotInOldContents: "v ~: old contents";
457             note NotInOldConFirst: "v ~: (fieldRead (old conts) first)" from
                    NotInOldConFirst, NotInOldContents, ContentsDef, ThisProps, Init, HProp;
458             note FirstNotN: "first ~= n";
459             note NewConDef: "theinv ConDef" from NewConDef, ConDef, NewNotRefByNext,
                    FirstNotN, NotInOldConFirst; */
460
461         /*: note ElemInj: "theinv ElementInjInv" from ElementInjInv, ThisProps,
                NewNotHT, NewNotRefArray, TableInjInv, ThisTableNotNull; */
462
463         {
464             /*: pickAny hs::obj; */
465             {
```

```
            /*: assuming h1: "hs : alloc & hs : HashSet & hs..init"; */
            {
                /*: pickAny i::int, w::obj;
                    note g1: "arrayRead (old arrayState) (hs..table) i ~= n";
                    note g2: "hs : old alloc"; */
                {
                    /*: assuming h2: "0 <= i & i < hs..table..length & w :
                        hs..table.[i]..conts"; */
                    {
                        /*: assuming h3: "hs = this";
                            note c5: "h w (hs..table..length) = i" from c3, h1, h2,
                                h3, Coherence, g1, g2, HProp; */
                    }
                    {
                        /*: assuming h4: "hs ~= this";
                            note g3: "hs..table ~= table";
                            note c4: "h w (hs..table..length) = i" from c4, h1, h2,
                                h4, Coherence, g1, g2, g3; */
                    }
                    /*: note c3: "h w (hs..table..length) = i" from c4, c5; */
                }
                /*: note c2: "0 <= i & i < hs..table..length --> w :
                    hs..table.[i]..conts --> h w (hs..table..length) = i" forSuch i,
                    w; */
            }
            /*: note c1: "ALL i w. 0 <= i & i < hs..table..length --> w :
                hs..table.[i]..conts --> h w (hs..table..length) = i"; */
        }
        /*: note CohPost: "hs : alloc & hs : HashSet & hs..init --> (ALL i w. 0 <=
            i & i < hs..table..length --> w : hs..table.[i]..conts --> h w
            (hs..table..length) = i)" forSuch hs; */
    }
    {
        /*: pickAny hs::obj; */
        {
            /*: assuming NewContentsHyp: "hs : alloc & hs : HashSet & hs..init";
                note ThisContents: "contents = {w. w : table.[(h w
                    (table..length))]..conts}" from ThisContents, HProp,
                    NewNotRefThisArr, HashInv, ContentsDefInv,  ThisProps, Init; */
            {
                /*: assuming OtherHyp: "hs ~= this";
                    note OtherContents: "hs..contents = {w. w : hs..table.[(h w
                        (hs..table..length))]..conts}" from OtherHyp,
                        NewContentsHyp, ContentsDefInvHash, NewNotHT,
                        TableInjInvHash, NewNotRefArray, TableNotNullHash, HashInv;
                        */
            }

            /*: cases "hs = this", "hs ~= this" for NewContentsCases: "hs..contents
                = {w. w : hs..table.[(h w (hs..table..length))]..conts}" from
                ThisContents, OtherContents;
                note "hs..contents = {w. w : hs..table.[(h w
                    (hs..table..length))]..conts}"; */
        }
        /*: note NewContentsDef: "hs : alloc & hs : HashSet & hs..init -->
            hs..contents = {w. w : hs..table.[(h w (hs..table..length))]..conts}"
            forSuch hs; */
    }

    /*: note OldNotRefByNext: "ALL x. first ~= null --> old (x..next) ~= first"
        from FirstInjInv, Init, HCBounds, OldNotRefByNext, ThisProps;

        note NewFirstInj: "theinv FirstInjInv" from FirstInjInv, NewNotRefByNext,
            OldNotRefByNext, TableInjInv, OldNotRefInTable, HCBounds, NewFirstInj,
            ThisProps, ElementInjInv, NewNotHT;
```

```
509
510             note NewNextInj: "theinv NextInjInv" from NextInjInv, OldNotRefByNext,
                    NewNextInj; */
511
512         /*: note ContentsPost: "contents = old contents Un {v}" */
513         {
514             /*: pickAny x::obj */
515             {
516                 /*: assuming CardHyp: "x : alloc & x : HashSet & x..init" */
517                 {
518                     /*: note ThisProps: "this : old alloc & this : HashSet & old init"
                            */
519                     /*: note OldCard: "old _size = card (old contents)" from ThisProps,
                            CardInv */
520                     /*: note NewLength: "_size = old _size + 1" */
521                     /*: note NewNotInOld: "v ~: old contents" */
522                     /*: note XIsThisCard: "_size = card (contents)" from OldCard,
                            NewLength, NewNotInOld, ContentsPost */
523                 }
524                 {
525                     /*: assuming XNotThisHyp: "x ~= this" */
526                     /*: note XInOldAlloc: "x : old alloc" */
527                     /*: note XOldInit: "x..(old HashSet.init)" */
528                     /*: note OldCard: "x..(old HashSet._size) = card (x..(old
                            HashSet.contents))" from CardHyp, XNotThisHyp, XInOldAlloc,
                            XOldInit, CardInv */
529                     /*: note XLengthEq: "x.._size = x..(old HashSet._size)" */
530                     {
531                         /*: localize */
532                         /*: note XContentsForw: "ALL y. y : x..contents --> y : x..(old
                                HashSet.contents)" */
533                         /*: note XContentsBack: "ALL y. y : x..(old HashSet.contents)
                                --> y : x..contents" */
534                         /*: note XContentsUnchanged: "x..contents = x..(old
                                HashSet.contents)" from XContentsForw, XContentsBack */
535                     }
536                     /*: note XNotThisCard: "x.._size = card (x..contents)" from
                            XLengthEq, OldCard, XContentsUnchanged */
537                 }
538                 /*: note CardConc: "x.._size = card (x..contents)" from XIsThisCard,
                        XNotThisCard */
539             }
540             /*: note CardPostCond: "x : alloc & x : HashSet & x..init --> x.._size =
                    card (x..contents)" forSuch x */
541         }
542     }
543
544     public boolean add(Object v)
545     /*: requires "init & v ~= null"
546         modifies "contents", "size"
547         ensures "(v ~: old contents --> contents = old contents Un {v} & size = old
                size + 1 & result) & (v : old contents --> contents = old contents & size =
                old size & ~result)" */
548     {
549         if (_contains(v)) {
550             return false;
551         } else {
552             _add(v);
553             return true;
554         }
555     }
556
557     public int size()
558     /*: requires "init"
559         ensures "result = size" */
```

```
    {
        return _size;
    }
}
```

B.3.2 Commutativity Testing Methods

Listing 8. HashSetComm.java

```
class HashSetComm {
    static void add_add_pre_s_0(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v1 ~= v2 | v1 : sa..contents" */
        boolean r1a = sa.add(v1);
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        boolean r1b = sb.add(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void add_add_pre_c_0(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 ~= v2 | v1 : sa..contents)" */
        boolean r1a = sa.add(v1);
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        boolean r1b = sb.add(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void add_add_between_s_1(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.add(v1);
        /*: assume "v1 ~= v2 | ~r1a" */
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        boolean r1b = sb.add(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void add_add_between_c_1(HashSet sa, HashSet sb, Object v1, Object v2)
```

```
51      /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
52                  sa..contents = sb..contents & sa..size = sb..size"
53          modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
54          ensures "True" */
55      {
56          boolean r1a = sa.add(v1);
57          /*: assume "~(v1 ~= v2 | ~r1a)" */
58          boolean r2a = sa.add(v2);
59
60          boolean r2b = sb.add(v2);
61          boolean r1b = sb.add(v1);
62
63          /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
64      }
65
66      static void add_add_post_s_2(HashSet sa, HashSet sb, Object v1, Object v2)
67      /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
68                  sa..contents = sb..contents & sa..size = sb..size"
69          modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
70          ensures "True" */
71      {
72          boolean r1a = sa.add(v1);
73          boolean r2a = sa.add(v2);
74          /*: assume "v1 ~= v2 | ~r1a" */
75
76          boolean r2b = sb.add(v2);
77          boolean r1b = sb.add(v1);
78
79          /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
80      }
81
82      static void add_add_post_c_2(HashSet sa, HashSet sb, Object v1, Object v2)
83      /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
84                  sa..contents = sb..contents & sa..size = sb..size"
85          modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
86          ensures "True" */
87      {
88          boolean r1a = sa.add(v1);
89          boolean r2a = sa.add(v2);
90          /*: assume "~(v1 ~= v2 | ~r1a)" */
91
92          boolean r2b = sb.add(v2);
93          boolean r1b = sb.add(v1);
94
95          /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
96      }
97
98      static void add_add_pre_s_3(HashSet sa, HashSet sb, Object v1, Object v2)
99      /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
100                 sa..contents = sb..contents & sa..size = sb..size"
101         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
102         ensures "True" */
103     {
104         /*: assume "v1 ~= v2 | v1 : sa..contents" */
105         boolean r1a = sa.add(v1);
106         sa.add(v2);
107
108         sb.add(v2);
```

```
109         boolean r1b = sb.add(v1);
110
111         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
112     }
113
114     static void add_add_pre_c_3(HashSet sa, HashSet sb, Object v1, Object v2)
115     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
116                 sa..contents = sb..contents & sa..size = sb..size"
117         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
118         ensures "True" */
119     {
120         /*: assume "~(v1 ~= v2 | v1 : sa..contents)" */
121         boolean r1a = sa.add(v1);
122         sa.add(v2);
123
124         sb.add(v2);
125         boolean r1b = sb.add(v1);
126
127         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
128     }
129
130     static void add_add_between_s_4(HashSet sa, HashSet sb, Object v1, Object v2)
131     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
132                 sa..contents = sb..contents & sa..size = sb..size"
133         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
134         ensures "True" */
135     {
136         boolean r1a = sa.add(v1);
137         /*: assume "v1 ~= v2 | ~r1a" */
138         sa.add(v2);
139
140         sb.add(v2);
141         boolean r1b = sb.add(v1);
142
143         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
144     }
145
146     static void add_add_between_c_4(HashSet sa, HashSet sb, Object v1, Object v2)
147     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
148                 sa..contents = sb..contents & sa..size = sb..size"
149         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
150         ensures "True" */
151     {
152         boolean r1a = sa.add(v1);
153         /*: assume "~(v1 ~= v2 | ~r1a)" */
154         sa.add(v2);
155
156         sb.add(v2);
157         boolean r1b = sb.add(v1);
158
159         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
160     }
161
162     static void add_add_post_s_5(HashSet sa, HashSet sb, Object v1, Object v2)
163     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
164                 sa..contents = sb..contents & sa..size = sb..size"
165         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
166         ensures "True" */
167     {
168         boolean r1a = sa.add(v1);
169         sa.add(v2);
```

```
170         /*: assume "v1 ~= v2 | ~r1a" */
171
172         sb.add(v2);
173         boolean r1b = sb.add(v1);
174
175         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
176     }
177
178     static void add_add_post_c_5(HashSet sa, HashSet sb, Object v1, Object v2)
179     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
180                 sa..contents = sb..contents & sa..size = sb..size"
181         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
182         ensures "True" */
183     {
184         boolean r1a = sa.add(v1);
185         sa.add(v2);
186         /*: assume "~(v1 ~= v2 | ~r1a)" */
187
188         sb.add(v2);
189         boolean r1b = sb.add(v1);
190
191         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
192     }
193
194     static void add_contains_pre_s_6(HashSet sa, HashSet sb, Object v1, Object v2)
195     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
196                 sa..contents = sb..contents & sa..size = sb..size"
197         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
198         ensures "True" */
199     {
200         /*: assume "v1 ~= v2 | v1 : sa..contents" */
201         boolean r1a = sa.add(v1);
202         boolean r2a = sa.contains(v2);
203
204         boolean r2b = sb.contains(v2);
205         boolean r1b = sb.add(v1);
206
207         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
208     }
209
210     static void add_contains_pre_c_6(HashSet sa, HashSet sb, Object v1, Object v2)
211     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
212                 sa..contents = sb..contents & sa..size = sb..size"
213         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
214         ensures "True" */
215     {
216         /*: assume "~(v1 ~= v2 | v1 : sa..contents)" */
217         boolean r1a = sa.add(v1);
218         boolean r2a = sa.contains(v2);
219
220         boolean r2b = sb.contains(v2);
221         boolean r1b = sb.add(v1);
222
223         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
224     }
225
226     static void add_contains_between_s_7(HashSet sa, HashSet sb, Object v1, Object v2)
227     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
228                 sa..contents = sb..contents & sa..size = sb..size"
```

```
229         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
230         ensures "True" */
231     {
232         boolean r1a = sa.add(v1);
233         /*: assume "v1 ~= v2 | ~r1a" */
234         boolean r2a = sa.contains(v2);
235 
236         boolean r2b = sb.contains(v2);
237         boolean r1b = sb.add(v1);
238 
239         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
240     }
241 
242     static void add_contains_between_c_7(HashSet sa, HashSet sb, Object v1, Object v2)
243     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
244                     sa..contents = sb..contents & sa..size = sb..size"
245         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
246         ensures "True" */
247     {
248         boolean r1a = sa.add(v1);
249         /*: assume "~(v1 ~= v2 | ~r1a)" */
250         boolean r2a = sa.contains(v2);
251 
252         boolean r2b = sb.contains(v2);
253         boolean r1b = sb.add(v1);
254 
255         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
256     }
257 
258     static void add_contains_post_s_8(HashSet sa, HashSet sb, Object v1, Object v2)
259     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
260                     sa..contents = sb..contents & sa..size = sb..size"
261         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
262         ensures "True" */
263     {
264         boolean r1a = sa.add(v1);
265         boolean r2a = sa.contains(v2);
266         /*: assume "v1 ~= v2 | ~r1a" */
267 
268         boolean r2b = sb.contains(v2);
269         boolean r1b = sb.add(v1);
270 
271         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
272     }
273 
274     static void add_contains_post_c_8(HashSet sa, HashSet sb, Object v1, Object v2)
275     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
276                     sa..contents = sb..contents & sa..size = sb..size"
277         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
278         ensures "True" */
279     {
280         boolean r1a = sa.add(v1);
281         boolean r2a = sa.contains(v2);
282         /*: assume "~(v1 ~= v2 | ~r1a)" */
283 
284         boolean r2b = sb.contains(v2);
285         boolean r1b = sb.add(v1);
286 
```

```
287         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
288     }
289
290     static void add_remove_pre_s_9(HashSet sa, HashSet sb, Object v1, Object v2)
291     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
292                 sa..contents = sb..contents & sa..size = sb..size"
293         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
294         ensures "True" */
295     {
296         /*: assume "v1 ~= v2" */
297         boolean r1a = sa.add(v1);
298         boolean r2a = sa.remove(v2);
299
300         boolean r2b = sb.remove(v2);
301         boolean r1b = sb.add(v1);
302
303         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
304     }
305
306     static void add_remove_pre_c_9(HashSet sa, HashSet sb, Object v1, Object v2)
307     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
308                 sa..contents = sb..contents & sa..size = sb..size"
309         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
310         ensures "True" */
311     {
312         /*: assume "~(v1 ~= v2)" */
313         boolean r1a = sa.add(v1);
314         boolean r2a = sa.remove(v2);
315
316         boolean r2b = sb.remove(v2);
317         boolean r1b = sb.add(v1);
318
319         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
320     }
321
322     static void add_remove_between_s_10(HashSet sa, HashSet sb, Object v1, Object v2)
323     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
324                 sa..contents = sb..contents & sa..size = sb..size"
325         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
326         ensures "True" */
327     {
328         boolean r1a = sa.add(v1);
329         /*: assume "v1 ~= v2" */
330         boolean r2a = sa.remove(v2);
331
332         boolean r2b = sb.remove(v2);
333         boolean r1b = sb.add(v1);
334
335         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
336     }
337
338     static void add_remove_between_c_10(HashSet sa, HashSet sb, Object v1, Object v2)
339     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
340                 sa..contents = sb..contents & sa..size = sb..size"
341         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
342         ensures "True" */
343     {
```

```
344         boolean r1a = sa.add(v1);
345         /*: assume "~(v1 ~= v2)" */
346         boolean r2a = sa.remove(v2);
347 
348         boolean r2b = sb.remove(v2);
349         boolean r1b = sb.add(v1);
350 
351         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
352     }
353 
354     static void add_remove_post_s_11(HashSet sa, HashSet sb, Object v1, Object v2)
355     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
356                 sa..contents = sb..contents & sa..size = sb..size"
357         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
358         ensures "True" */
359     {
360         boolean r1a = sa.add(v1);
361         boolean r2a = sa.remove(v2);
362         /*: assume "v1 ~= v2" */
363 
364         boolean r2b = sb.remove(v2);
365         boolean r1b = sb.add(v1);
366 
367         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
368     }
369 
370     static void add_remove_post_c_11(HashSet sa, HashSet sb, Object v1, Object v2)
371     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
372                 sa..contents = sb..contents & sa..size = sb..size"
373         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
374         ensures "True" */
375     {
376         boolean r1a = sa.add(v1);
377         boolean r2a = sa.remove(v2);
378         /*: assume "~(v1 ~= v2)" */
379 
380         boolean r2b = sb.remove(v2);
381         boolean r1b = sb.add(v1);
382 
383         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
384     }
385 
386     static void add_remove_pre_s_12(HashSet sa, HashSet sb, Object v1, Object v2)
387     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
388                 sa..contents = sb..contents & sa..size = sb..size"
389         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
390         ensures "True" */
391     {
392         /*: assume "v1 ~= v2" */
393         boolean r1a = sa.add(v1);
394         sa.remove(v2);
395 
396         sb.remove(v2);
397         boolean r1b = sb.add(v1);
398 
399         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
400     }
401 
402     static void add_remove_pre_c_12(HashSet sa, HashSet sb, Object v1, Object v2)
```

```
403     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
404                 sa..contents = sb..contents & sa..size = sb..size"
405         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
406         ensures "True" */
407     {
408         /*: assume "~(v1 ~= v2)" */
409         boolean r1a = sa.add(v1);
410         sa.remove(v2);
411
412         sb.remove(v2);
413         boolean r1b = sb.add(v1);
414
415         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
416     }
417
418     static void add_remove_between_s_13(HashSet sa, HashSet sb, Object v1, Object v2)
419     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
420                 sa..contents = sb..contents & sa..size = sb..size"
421         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
422         ensures "True" */
423     {
424         boolean r1a = sa.add(v1);
425         /*: assume "v1 ~= v2" */
426         sa.remove(v2);
427
428         sb.remove(v2);
429         boolean r1b = sb.add(v1);
430
431         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
432     }
433
434     static void add_remove_between_c_13(HashSet sa, HashSet sb, Object v1, Object v2)
435     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
436                 sa..contents = sb..contents & sa..size = sb..size"
437         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
438         ensures "True" */
439     {
440         boolean r1a = sa.add(v1);
441         /*: assume "~(v1 ~= v2)" */
442         sa.remove(v2);
443
444         sb.remove(v2);
445         boolean r1b = sb.add(v1);
446
447         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
448     }
449
450     static void add_remove_post_s_14(HashSet sa, HashSet sb, Object v1, Object v2)
451     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
452                 sa..contents = sb..contents & sa..size = sb..size"
453         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
454         ensures "True" */
455     {
456         boolean r1a = sa.add(v1);
457         sa.remove(v2);
458         /*: assume "v1 ~= v2" */
459
460         sb.remove(v2);
461         boolean r1b = sb.add(v1);
462
463         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
```

```
464     }
465
466     static void add_remove_post_c_14(HashSet sa, HashSet sb, Object v1, Object v2)
467     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
468                    sa..contents = sb..contents & sa..size = sb..size"
469         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
470         ensures "True" */
471     {
472         boolean r1a = sa.add(v1);
473         sa.remove(v2);
474         /*: assume "~(v1 ~= v2)" */
475
476         sb.remove(v2);
477         boolean r1b = sb.add(v1);
478
479         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
480     }
481
482     static void add_size_pre_s_15(HashSet sa, HashSet sb, Object v1)
483     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            &
484                    sa..contents = sb..contents & sa..size = sb..size"
485         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
486         ensures "True" */
487     {
488         /*: assume "v1 : sa..contents" */
489         boolean r1a = sa.add(v1);
490         int r2a = sa.size();
491
492         int r2b = sb.size();
493         boolean r1b = sb.add(v1);
494
495         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
496     }
497
498     static void add_size_pre_c_15(HashSet sa, HashSet sb, Object v1)
499     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            &
500                    sa..contents = sb..contents & sa..size = sb..size"
501         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
502         ensures "True" */
503     {
504         /*: assume "~(v1 : sa..contents)" */
505         boolean r1a = sa.add(v1);
506         int r2a = sa.size();
507
508         int r2b = sb.size();
509         boolean r1b = sb.add(v1);
510
511         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
512     }
513
514     static void add_size_between_s_16(HashSet sa, HashSet sb, Object v1)
515     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            &
516                    sa..contents = sb..contents & sa..size = sb..size"
517         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
518         ensures "True" */
519     {
520         boolean r1a = sa.add(v1);
521         /*: assume "~r1a" */
522         int r2a = sa.size();
```

```
523
524         int r2b = sb.size();
525         boolean r1b = sb.add(v1);
526
527         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
528     }
529
530     static void add_size_between_c_16(HashSet sa, HashSet sb, Object v1)
531     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            &
532                 sa..contents = sb..contents & sa..size = sb..size"
533         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
534         ensures "True" */
535     {
536         boolean r1a = sa.add(v1);
537         /*: assume "~(~r1a)" */
538         int r2a = sa.size();
539
540         int r2b = sb.size();
541         boolean r1b = sb.add(v1);
542
543         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
544     }
545
546     static void add_size_post_s_17(HashSet sa, HashSet sb, Object v1)
547     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            &
548                 sa..contents = sb..contents & sa..size = sb..size"
549         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
550         ensures "True" */
551     {
552         boolean r1a = sa.add(v1);
553         int r2a = sa.size();
554         /*: assume "~r1a" */
555
556         int r2b = sb.size();
557         boolean r1b = sb.add(v1);
558
559         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
560     }
561
562     static void add_size_post_c_17(HashSet sa, HashSet sb, Object v1)
563     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            &
564                 sa..contents = sb..contents & sa..size = sb..size"
565         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
566         ensures "True" */
567     {
568         boolean r1a = sa.add(v1);
569         int r2a = sa.size();
570         /*: assume "~(~r1a)" */
571
572         int r2b = sb.size();
573         boolean r1b = sb.add(v1);
574
575         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
576     }
577
578     static void add_add_pre_s_18(HashSet sa, HashSet sb, Object v1, Object v2)
579     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
```

```
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v1 ~= v2 | v1 : sa..contents" */
        sa.add(v1);
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        sb.add(v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void add_add_pre_c_18(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 ~= v2 | v1 : sa..contents)" */
        sa.add(v1);
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        sb.add(v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void add_add_between_s_19(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.add(v1);
        /*: assume "v1 ~= v2 | v1 : sa..(old contents)" */
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        sb.add(v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void add_add_between_c_19(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.add(v1);
        /*: assume "~(v1 ~= v2 | v1 : sa..(old contents))" */
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        sb.add(v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

```

```
642     static void add_add_post_s_20(HashSet sa, HashSet sb, Object v1, Object v2)
643     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
644                 sa..contents = sb..contents & sa..size = sb..size"
645         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
646         ensures "True" */
647     {
648         sa.add(v1);
649         boolean r2a = sa.add(v2);
650         /*: assume "v1 ~= v2 | v1 : sa..(old contents)" */
651 
652         boolean r2b = sb.add(v2);
653         sb.add(v1);
654 
655         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
656     }
657 
658     static void add_add_post_c_20(HashSet sa, HashSet sb, Object v1, Object v2)
659     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
660                 sa..contents = sb..contents & sa..size = sb..size"
661         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
662         ensures "True" */
663     {
664         sa.add(v1);
665         boolean r2a = sa.add(v2);
666         /*: assume "~(v1 ~= v2 | v1 : sa..(old contents))" */
667 
668         boolean r2b = sb.add(v2);
669         sb.add(v1);
670 
671         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
672     }
673 
674     static void add_add_pre_s_21(HashSet sa, HashSet sb, Object v1, Object v2)
675     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
676                 sa..contents = sb..contents & sa..size = sb..size"
677         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
678         ensures "True" */
679     {
680         /*: assume "True" */
681         sa.add(v1);
682         sa.add(v2);
683 
684         sb.add(v2);
685         sb.add(v1);
686 
687         /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
688     }
689 
690     static void add_add_pre_c_21(HashSet sa, HashSet sb, Object v1, Object v2)
691     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
692                 sa..contents = sb..contents & sa..size = sb..size"
693         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
694         ensures "True" */
695     {
696         /*: assume "~(True)" */
697         sa.add(v1);
698         sa.add(v2);
699 
700         sb.add(v2);
701         sb.add(v1);
702 
```

```
        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void add_add_between_s_22(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.add(v1);
        /*: assume "True" */
        sa.add(v2);

        sb.add(v2);
        sb.add(v1);

        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void add_add_between_c_22(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.add(v1);
        /*: assume "~(True)" */
        sa.add(v2);

        sb.add(v2);
        sb.add(v1);

        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void add_add_post_s_23(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.add(v1);
        sa.add(v2);
        /*: assume "True" */

        sb.add(v2);
        sb.add(v1);

        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void add_add_post_c_23(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.add(v1);
        sa.add(v2);
        /*: assume "~(True)" */

```

```
764         sb.add(v2);
765         sb.add(v1);
766 
767         /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
768     }
769 
770     static void add_contains_pre_s_24(HashSet sa, HashSet sb, Object v1, Object v2)
771     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
772                 sa..contents = sb..contents & sa..size = sb..size"
773         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
774         ensures "True" */
775     {
776         /*: assume "v1 ~= v2 | v1 : sa..contents" */
777         sa.add(v1);
778         boolean r2a = sa.contains(v2);
779 
780         boolean r2b = sb.contains(v2);
781         sb.add(v1);
782 
783         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
784     }
785 
786     static void add_contains_pre_c_24(HashSet sa, HashSet sb, Object v1, Object v2)
787     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
788                 sa..contents = sb..contents & sa..size = sb..size"
789         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
790         ensures "True" */
791     {
792         /*: assume "~(v1 ~= v2 | v1 : sa..contents)" */
793         sa.add(v1);
794         boolean r2a = sa.contains(v2);
795 
796         boolean r2b = sb.contains(v2);
797         sb.add(v1);
798 
799         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
800     }
801 
802     static void add_contains_between_s_25(HashSet sa, HashSet sb, Object v1, Object v2)
803     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
804                 sa..contents = sb..contents & sa..size = sb..size"
805         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
806         ensures "True" */
807     {
808         sa.add(v1);
809         /*: assume "v1 ~= v2 | v1 : sa..(old contents)" */
810         boolean r2a = sa.contains(v2);
811 
812         boolean r2b = sb.contains(v2);
813         sb.add(v1);
814 
815         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
816     }
817 
818     static void add_contains_between_c_25(HashSet sa, HashSet sb, Object v1, Object v2)
819     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
820                 sa..contents = sb..contents & sa..size = sb..size"
821         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
822         ensures "True" */
823     {
824         sa.add(v1);
```

```
825         /*: assume "~(v1 ~= v2 | v1 : sa..(old contents))" */
826         boolean r2a = sa.contains(v2);
827 
828         boolean r2b = sb.contains(v2);
829         sb.add(v1);
830 
831         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
832     }
833 
834     static void add_contains_post_s_26(HashSet sa, HashSet sb, Object v1, Object v2)
835     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
836                 sa..contents = sb..contents & sa..size = sb..size"
837         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
838         ensures "True" */
839     {
840         sa.add(v1);
841         boolean r2a = sa.contains(v2);
842         /*: assume "v1 ~= v2 | v1 : sa..(old contents)" */
843 
844         boolean r2b = sb.contains(v2);
845         sb.add(v1);
846 
847         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
848     }
849 
850     static void add_contains_post_c_26(HashSet sa, HashSet sb, Object v1, Object v2)
851     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
852                 sa..contents = sb..contents & sa..size = sb..size"
853         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
854         ensures "True" */
855     {
856         sa.add(v1);
857         boolean r2a = sa.contains(v2);
858         /*: assume "~(v1 ~= v2 | v1 : sa..(old contents))" */
859 
860         boolean r2b = sb.contains(v2);
861         sb.add(v1);
862 
863         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
864     }
865 
866     static void add_remove_pre_s_27(HashSet sa, HashSet sb, Object v1, Object v2)
867     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
868                 sa..contents = sb..contents & sa..size = sb..size"
869         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
870         ensures "True" */
871     {
872         /*: assume "v1 ~= v2" */
873         sa.add(v1);
874         boolean r2a = sa.remove(v2);
875 
876         boolean r2b = sb.remove(v2);
877         sb.add(v1);
878 
879         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
880     }
881 
882     static void add_remove_pre_c_27(HashSet sa, HashSet sb, Object v1, Object v2)
883     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
884                 sa..contents = sb..contents & sa..size = sb..size"
885         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
```

```
886         ensures "True" */
887     {
888         /*: assume "~(v1 ~= v2)" */
889         sa.add(v1);
890         boolean r2a = sa.remove(v2);
891 
892         boolean r2b = sb.remove(v2);
893         sb.add(v1);
894 
895         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
896     }
897 
898     static void add_remove_between_s_28(HashSet sa, HashSet sb, Object v1, Object v2)
899     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
900                 sa..contents = sb..contents & sa..size = sb..size"
901         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
902         ensures "True" */
903     {
904         sa.add(v1);
905         /*: assume "v1 ~= v2" */
906         boolean r2a = sa.remove(v2);
907 
908         boolean r2b = sb.remove(v2);
909         sb.add(v1);
910 
911         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
912     }
913 
914     static void add_remove_between_c_28(HashSet sa, HashSet sb, Object v1, Object v2)
915     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
916                 sa..contents = sb..contents & sa..size = sb..size"
917         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
918         ensures "True" */
919     {
920         sa.add(v1);
921         /*: assume "~(v1 ~= v2)" */
922         boolean r2a = sa.remove(v2);
923 
924         boolean r2b = sb.remove(v2);
925         sb.add(v1);
926 
927         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
928     }
929 
930     static void add_remove_post_s_29(HashSet sa, HashSet sb, Object v1, Object v2)
931     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
932                 sa..contents = sb..contents & sa..size = sb..size"
933         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
934         ensures "True" */
935     {
936         sa.add(v1);
937         boolean r2a = sa.remove(v2);
938         /*: assume "v1 ~= v2" */
939 
940         boolean r2b = sb.remove(v2);
941         sb.add(v1);
942 
943         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
944     }
945 
946     static void add_remove_post_c_29(HashSet sa, HashSet sb, Object v1, Object v2)
```

```
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.add(v1);
        boolean r2a = sa.remove(v2);
        /*: assume "~(v1 ~= v2)" */

        boolean r2b = sb.remove(v2);
        sb.add(v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void add_remove_pre_s_30(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v1 ~= v2" */
        sa.add(v1);
        sa.remove(v2);

        sb.remove(v2);
        sb.add(v1);

        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void add_remove_pre_c_30(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 ~= v2)" */
        sa.add(v1);
        sa.remove(v2);

        sb.remove(v2);
        sb.add(v1);

        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void add_remove_between_s_31(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.add(v1);
        /*: assume "v1 ~= v2" */
        sa.remove(v2);

        sb.remove(v2);
        sb.add(v1);

        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
```

```
1008    }
1009
1010    static void add_remove_between_c_31(HashSet sa, HashSet sb, Object v1, Object v2)
1011    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
1012                sa..contents = sb..contents & sa..size = sb..size"
1013        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1014        ensures "True" */
1015    {
1016        sa.add(v1);
1017        /*: assume "~(v1 ~= v2)" */
1018        sa.remove(v2);
1019
1020        sb.remove(v2);
1021        sb.add(v1);
1022
1023        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
1024    }
1025
1026    static void add_remove_post_s_32(HashSet sa, HashSet sb, Object v1, Object v2)
1027    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
1028                sa..contents = sb..contents & sa..size = sb..size"
1029        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1030        ensures "True" */
1031    {
1032        sa.add(v1);
1033        sa.remove(v2);
1034        /*: assume "v1 ~= v2" */
1035
1036        sb.remove(v2);
1037        sb.add(v1);
1038
1039        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
1040    }
1041
1042    static void add_remove_post_c_32(HashSet sa, HashSet sb, Object v1, Object v2)
1043    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
1044                sa..contents = sb..contents & sa..size = sb..size"
1045        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1046        ensures "True" */
1047    {
1048        sa.add(v1);
1049        sa.remove(v2);
1050        /*: assume "~(v1 ~= v2)" */
1051
1052        sb.remove(v2);
1053        sb.add(v1);
1054
1055        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
1056    }
1057
1058    static void add_size_pre_s_33(HashSet sa, HashSet sb, Object v1)
1059    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            &
1060                sa..contents = sb..contents & sa..size = sb..size"
1061        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1062        ensures "True" */
1063    {
1064        /*: assume "v1 : sa..contents" */
1065        sa.add(v1);
1066        int r2a = sa.size();
1067
1068        int r2b = sb.size();
```

```
        sb.add(v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void add_size_pre_c_33(HashSet sa, HashSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 : sa..contents)" */
        sa.add(v1);
        int r2a = sa.size();

        int r2b = sb.size();
        sb.add(v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void add_size_between_s_34(HashSet sa, HashSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.add(v1);
        /*: assume "v1 : sa..(old contents)" */
        int r2a = sa.size();

        int r2b = sb.size();
        sb.add(v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void add_size_between_c_34(HashSet sa, HashSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.add(v1);
        /*: assume "~(v1 : sa..(old contents))" */
        int r2a = sa.size();

        int r2b = sb.size();
        sb.add(v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void add_size_post_s_35(HashSet sa, HashSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.add(v1);
        int r2a = sa.size();
```

```
        /*: assume "v1 : sa..(old contents)" */

        int r2b = sb.size();
        sb.add(v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void add_size_post_c_35(HashSet sa, HashSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.add(v1);
        int r2a = sa.size();
        /*: assume "~(v1 : sa..(old contents))" */

        int r2b = sb.size();
        sb.add(v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void contains_add_pre_s_36(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v1 ~= v2 | v1 : sa..contents" */
        boolean r1a = sa.contains(v1);
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void contains_add_pre_c_36(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 ~= v2 | v1 : sa..contents)" */
        boolean r1a = sa.contains(v1);
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void contains_add_between_s_37(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                   sa..contents = sb..contents & sa..size = sb..size"
```

```
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        /*: assume "v1 ~= v2 | r1a" */
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void contains_add_between_c_37(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        /*: assume "~(v1 ~= v2 | r1a)" */
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void contains_add_post_s_38(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        boolean r2a = sa.add(v2);
        /*: assume "v1 ~= v2 | r1a" */

        boolean r2b = sb.add(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void contains_add_post_c_38(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        boolean r2a = sa.add(v2);
        /*: assume "~(v1 ~= v2 | r1a)" */

        boolean r2b = sb.add(v2);
        boolean r1b = sb.contains(v1);

```

```
        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void contains_add_pre_s_39(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v1 ~= v2 | v1 : sa..contents" */
        boolean r1a = sa.contains(v1);
        sa.add(v2);

        sb.add(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void contains_add_pre_c_39(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 ~= v2 | v1 : sa..contents)" */
        boolean r1a = sa.contains(v1);
        sa.add(v2);

        sb.add(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void contains_add_between_s_40(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        /*: assume "v1 ~= v2 | r1a" */
        sa.add(v2);

        sb.add(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void contains_add_between_c_40(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        /*: assume "~(v1 ~= v2 | r1a)" */
        sa.add(v2);
```

```
        sb.add(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void contains_add_post_s_41(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        sa.add(v2);
        /*: assume "v1 ~= v2 | r1a" */

        sb.add(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void contains_add_post_c_41(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        sa.add(v2);
        /*: assume "~(v1 ~= v2 | r1a)" */

        sb.add(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void contains_contains_pre_s_42(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        /*: assume "True" */
        boolean r1a = sa.contains(v1);
        boolean r2a = sa.contains(v2);

        boolean r2b = sb.contains(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void contains_contains_pre_c_42(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        /*: assume "~(True)" */
```

```
1367        boolean r1a = sa.contains(v1);
1368        boolean r2a = sa.contains(v2);
1369
1370        boolean r2b = sb.contains(v2);
1371        boolean r1b = sb.contains(v1);
1372
1373        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
1374    }
1375
1376    static void contains_contains_between_s_43(HashSet sa, HashSet sb, Object v1,
            Object v2)
1377    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
1378                sa..contents = sb..contents & sa..size = sb..size"
1379        ensures "True" */
1380    {
1381        boolean r1a = sa.contains(v1);
1382        /*: assume "True" */
1383        boolean r2a = sa.contains(v2);
1384
1385        boolean r2b = sb.contains(v2);
1386        boolean r1b = sb.contains(v1);
1387
1388        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
1389    }
1390
1391    static void contains_contains_between_c_43(HashSet sa, HashSet sb, Object v1,
            Object v2)
1392    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
1393                sa..contents = sb..contents & sa..size = sb..size"
1394        ensures "True" */
1395    {
1396        boolean r1a = sa.contains(v1);
1397        /*: assume "~(True)" */
1398        boolean r2a = sa.contains(v2);
1399
1400        boolean r2b = sb.contains(v2);
1401        boolean r1b = sb.contains(v1);
1402
1403        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
1404    }
1405
1406    static void contains_contains_post_s_44(HashSet sa, HashSet sb, Object v1, Object
            v2)
1407    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
1408                sa..contents = sb..contents & sa..size = sb..size"
1409        ensures "True" */
1410    {
1411        boolean r1a = sa.contains(v1);
1412        boolean r2a = sa.contains(v2);
1413        /*: assume "True" */
1414
1415        boolean r2b = sb.contains(v2);
1416        boolean r1b = sb.contains(v1);
1417
1418        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
1419    }
1420
```

```
    static void contains_contains_post_c_44(HashSet sa, HashSet sb, Object v1, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        boolean r2a = sa.contains(v2);
        /*: assume "~(True)" */

        boolean r2b = sb.contains(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void contains_remove_pre_s_45(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v1 ~= v2 | v1 ~: sa..contents" */
        boolean r1a = sa.contains(v1);
        boolean r2a = sa.remove(v2);

        boolean r2b = sb.remove(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void contains_remove_pre_c_45(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 ~= v2 | v1 ~: sa..contents)" */
        boolean r1a = sa.contains(v1);
        boolean r2a = sa.remove(v2);

        boolean r2b = sb.remove(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void contains_remove_between_s_46(HashSet sa, HashSet sb, Object v1, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        /*: assume "v1 ~= v2 | ~r1a" */
        boolean r2a = sa.remove(v2);
```

```
1477
1478            boolean r2b = sb.remove(v2);
1479            boolean r1b = sb.contains(v1);
1480
1481            /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                    sb..size" */
1482        }
1483
1484        static void contains_remove_between_c_46(HashSet sa, HashSet sb, Object v1, Object
                v2)
1485        /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
                & v2 ~= null &
1486                    sa..contents = sb..contents & sa..size = sb..size"
1487            modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1488            ensures "True" */
1489        {
1490            boolean r1a = sa.contains(v1);
1491            /*: assume "~(v1 ~= v2 | ~r1a)" */
1492            boolean r2a = sa.remove(v2);
1493
1494            boolean r2b = sb.remove(v2);
1495            boolean r1b = sb.contains(v1);
1496
1497            /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                    sb..size)" */
1498        }
1499
1500        static void contains_remove_post_s_47(HashSet sa, HashSet sb, Object v1, Object v2)
1501        /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
                & v2 ~= null &
1502                    sa..contents = sb..contents & sa..size = sb..size"
1503            modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1504            ensures "True" */
1505        {
1506            boolean r1a = sa.contains(v1);
1507            boolean r2a = sa.remove(v2);
1508            /*: assume "v1 ~= v2 | ~r1a" */
1509
1510            boolean r2b = sb.remove(v2);
1511            boolean r1b = sb.contains(v1);
1512
1513            /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                    sb..size" */
1514        }
1515
1516        static void contains_remove_post_c_47(HashSet sa, HashSet sb, Object v1, Object v2)
1517        /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
                & v2 ~= null &
1518                    sa..contents = sb..contents & sa..size = sb..size"
1519            modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1520            ensures "True" */
1521        {
1522            boolean r1a = sa.contains(v1);
1523            boolean r2a = sa.remove(v2);
1524            /*: assume "~(v1 ~= v2 | ~r1a)" */
1525
1526            boolean r2b = sb.remove(v2);
1527            boolean r1b = sb.contains(v1);
1528
1529            /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                    sb..size)" */
1530        }
1531
1532        static void contains_remove_pre_s_48(HashSet sa, HashSet sb, Object v1, Object v2)
```

```
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v1 ~= v2 | v1 ~: sa..contents" */
        boolean r1a = sa.contains(v1);
        sa.remove(v2);

        sb.remove(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void contains_remove_pre_c_48(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 ~= v2 | v1 ~: sa..contents)" */
        boolean r1a = sa.contains(v1);
        sa.remove(v2);

        sb.remove(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void contains_remove_between_s_49(HashSet sa, HashSet sb, Object v1, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        /*: assume "v1 ~= v2 | ~r1a" */
        sa.remove(v2);

        sb.remove(v2);
        boolean r1b = sb.contains(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void contains_remove_between_c_49(HashSet sa, HashSet sb, Object v1, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        /*: assume "~(v1 ~= v2 | ~r1a)" */
        sa.remove(v2);

        sb.remove(v2);
        boolean r1b = sb.contains(v1);
```

```
1592
1593        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1594    }
1595
1596    static void contains_remove_post_s_50(HashSet sa, HashSet sb, Object v1, Object v2)
1597    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
1598                sa..contents = sb..contents & sa..size = sb..size"
1599     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1600     ensures "True" */
1601    {
1602     boolean r1a = sa.contains(v1);
1603     sa.remove(v2);
1604     /*: assume "v1 ~= v2 | ~r1a" */
1605
1606     sb.remove(v2);
1607     boolean r1b = sb.contains(v1);
1608
1609     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1610    }
1611
1612    static void contains_remove_post_c_50(HashSet sa, HashSet sb, Object v1, Object v2)
1613    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
1614                sa..contents = sb..contents & sa..size = sb..size"
1615     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1616     ensures "True" */
1617    {
1618     boolean r1a = sa.contains(v1);
1619     sa.remove(v2);
1620     /*: assume "~(v1 ~= v2 | ~r1a)" */
1621
1622     sb.remove(v2);
1623     boolean r1b = sb.contains(v1);
1624
1625     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1626    }
1627
1628    static void contains_size_pre_s_51(HashSet sa, HashSet sb, Object v1)
1629    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        &
1630                sa..contents = sb..contents & sa..size = sb..size"
1631     ensures "True" */
1632    {
1633     /*: assume "True" */
1634     boolean r1a = sa.contains(v1);
1635     int r2a = sa.size();
1636
1637     int r2b = sb.size();
1638     boolean r1b = sb.contains(v1);
1639
1640     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
1641    }
1642
1643    static void contains_size_pre_c_51(HashSet sa, HashSet sb, Object v1)
1644    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        &
1645                sa..contents = sb..contents & sa..size = sb..size"
1646     ensures "True" */
1647    {
1648     /*: assume "~(True)" */
1649     boolean r1a = sa.contains(v1);
1650     int r2a = sa.size();
1651
```

```
        int r2b = sb.size();
        boolean r1b = sb.contains(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void contains_size_between_s_52(HashSet sa, HashSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        /*: assume "True" */
        int r2a = sa.size();

        int r2b = sb.size();
        boolean r1b = sb.contains(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void contains_size_between_c_52(HashSet sa, HashSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        /*: assume "~(True)" */
        int r2a = sa.size();

        int r2b = sb.size();
        boolean r1b = sb.contains(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void contains_size_post_s_53(HashSet sa, HashSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
        int r2a = sa.size();
        /*: assume "True" */

        int r2b = sb.size();
        boolean r1b = sb.contains(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void contains_size_post_c_53(HashSet sa, HashSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        boolean r1a = sa.contains(v1);
```

```
        int r2a = sa.size();
        /*: assume "~(True)" */

        int r2b = sb.size();
        boolean r1b = sb.contains(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void remove_add_pre_s_54(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v1 ~= v2" */
        boolean r1a = sa.remove(v1);
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void remove_add_pre_c_54(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 ~= v2)" */
        boolean r1a = sa.remove(v1);
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void remove_add_between_s_55(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        /*: assume "v1 ~= v2" */
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void remove_add_between_c_55(HashSet sa, HashSet sb, Object v1, Object v2)
```

```
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        /*: assume "~(v1 ~= v2)" */
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void remove_add_post_s_56(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        boolean r2a = sa.add(v2);
        /*: assume "v1 ~= v2" */

        boolean r2b = sb.add(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void remove_add_post_c_56(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        boolean r2a = sa.add(v2);
        /*: assume "~(v1 ~= v2)" */

        boolean r2b = sb.add(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void remove_add_pre_s_57(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v1 ~= v2" */
        boolean r1a = sa.remove(v1);
        sa.add(v2);

        sb.add(v2);
```

```
        boolean r1b = sb.remove(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_add_pre_c_57(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 ~= v2)" */
        boolean r1a = sa.remove(v1);
        sa.add(v2);

        sb.add(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_add_between_s_58(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        /*: assume "v1 ~= v2" */
        sa.add(v2);

        sb.add(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_add_between_c_58(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        /*: assume "~(v1 ~= v2)" */
        sa.add(v2);

        sb.add(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_add_post_s_59(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        sa.add(v2);
```

```
        /*: assume "v1 ~= v2" */

        sb.add(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_add_post_c_59(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        sa.add(v2);
        /*: assume "~(v1 ~= v2)" */

        sb.add(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_contains_pre_s_60(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v1 ~= v2 | v1 ~: sa..contents" */
        boolean r1a = sa.remove(v1);
        boolean r2a = sa.contains(v2);

        boolean r2b = sb.contains(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void remove_contains_pre_c_60(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 ~= v2 | v1 ~: sa..contents)" */
        boolean r1a = sa.remove(v1);
        boolean r2a = sa.contains(v2);

        boolean r2b = sb.contains(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void remove_contains_between_s_61(HashSet sa, HashSet sb, Object v1, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
```

```
1944                 sa..contents = sb..contents & sa..size = sb..size"
1945         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1946         ensures "True" */
1947     {
1948         boolean r1a = sa.remove(v1);
1949         /*: assume "v1 ~= v2 | ~r1a" */
1950         boolean r2a = sa.contains(v2);
1951 
1952         boolean r2b = sb.contains(v2);
1953         boolean r1b = sb.remove(v1);
1954 
1955         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
1956     }
1957 
1958     static void remove_contains_between_c_61(HashSet sa, HashSet sb, Object v1, Object
            v2)
1959     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
1960                 sa..contents = sb..contents & sa..size = sb..size"
1961         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1962         ensures "True" */
1963     {
1964         boolean r1a = sa.remove(v1);
1965         /*: assume "~(v1 ~= v2 | ~r1a)" */
1966         boolean r2a = sa.contains(v2);
1967 
1968         boolean r2b = sb.contains(v2);
1969         boolean r1b = sb.remove(v1);
1970 
1971         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
1972     }
1973 
1974     static void remove_contains_post_s_62(HashSet sa, HashSet sb, Object v1, Object v2)
1975     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
1976                 sa..contents = sb..contents & sa..size = sb..size"
1977         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1978         ensures "True" */
1979     {
1980         boolean r1a = sa.remove(v1);
1981         boolean r2a = sa.contains(v2);
1982         /*: assume "v1 ~= v2 | ~r1a" */
1983 
1984         boolean r2b = sb.contains(v2);
1985         boolean r1b = sb.remove(v1);
1986 
1987         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
1988     }
1989 
1990     static void remove_contains_post_c_62(HashSet sa, HashSet sb, Object v1, Object v2)
1991     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
1992                 sa..contents = sb..contents & sa..size = sb..size"
1993         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1994         ensures "True" */
1995     {
1996         boolean r1a = sa.remove(v1);
1997         boolean r2a = sa.contains(v2);
1998         /*: assume "~(v1 ~= v2 | ~r1a)" */
1999 
2000         boolean r2b = sb.contains(v2);
2001         boolean r1b = sb.remove(v1);
```

```
2002
2003        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
2004    }
2005
2006    static void remove_remove_pre_s_63(HashSet sa, HashSet sb, Object v1, Object v2)
2007    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
2008                    sa..contents = sb..contents & sa..size = sb..size"
2009        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2010        ensures "True" */
2011    {
2012        /*: assume "v1 ~= v2 | v1 ~: sa..contents" */
2013        boolean r1a = sa.remove(v1);
2014        boolean r2a = sa.remove(v2);
2015
2016        boolean r2b = sb.remove(v2);
2017        boolean r1b = sb.remove(v1);
2018
2019        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
2020    }
2021
2022    static void remove_remove_pre_c_63(HashSet sa, HashSet sb, Object v1, Object v2)
2023    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
2024                    sa..contents = sb..contents & sa..size = sb..size"
2025        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2026        ensures "True" */
2027    {
2028        /*: assume "~(v1 ~= v2 | v1 ~: sa..contents)" */
2029        boolean r1a = sa.remove(v1);
2030        boolean r2a = sa.remove(v2);
2031
2032        boolean r2b = sb.remove(v2);
2033        boolean r1b = sb.remove(v1);
2034
2035        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
2036    }
2037
2038    static void remove_remove_between_s_64(HashSet sa, HashSet sb, Object v1, Object v2)
2039    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
2040                    sa..contents = sb..contents & sa..size = sb..size"
2041        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2042        ensures "True" */
2043    {
2044        boolean r1a = sa.remove(v1);
2045        /*: assume "v1 ~= v2 | ~r1a" */
2046        boolean r2a = sa.remove(v2);
2047
2048        boolean r2b = sb.remove(v2);
2049        boolean r1b = sb.remove(v1);
2050
2051        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
2052    }
2053
2054    static void remove_remove_between_c_64(HashSet sa, HashSet sb, Object v1, Object v2)
2055    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
2056                    sa..contents = sb..contents & sa..size = sb..size"
2057        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2058        ensures "True" */
```

```
    {
        boolean r1a = sa.remove(v1);
        /*: assume "~(v1 ~= v2 | ~r1a)" */
        boolean r2a = sa.remove(v2);

        boolean r2b = sb.remove(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void remove_remove_post_s_65(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        boolean r2a = sa.remove(v2);
        /*: assume "v1 ~= v2 | ~r1a" */

        boolean r2b = sb.remove(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void remove_remove_post_c_65(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        boolean r2a = sa.remove(v2);
        /*: assume "~(v1 ~= v2 | ~r1a)" */

        boolean r2b = sb.remove(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void remove_remove_pre_s_66(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v1 ~= v2 | v1 ~: sa..contents" */
        boolean r1a = sa.remove(v1);
        sa.remove(v2);

        sb.remove(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }
```

```
    static void remove_remove_pre_c_66(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 ~= v2 | v1 ~: sa..contents)" */
        boolean r1a = sa.remove(v1);
        sa.remove(v2);

        sb.remove(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_remove_between_s_67(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        /*: assume "v1 ~= v2 | ~r1a" */
        sa.remove(v2);

        sb.remove(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_remove_between_c_67(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        /*: assume "~(v1 ~= v2 | ~r1a)" */
        sa.remove(v2);

        sb.remove(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_remove_post_s_68(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        sa.remove(v2);
        /*: assume "v1 ~= v2 | ~r1a" */

        sb.remove(v2);
        boolean r1b = sb.remove(v1);

```

```
        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_remove_post_c_68(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        sa.remove(v2);
        /*: assume "~(v1 ~= v2 | ~r1a)" */

        sb.remove(v2);
        boolean r1b = sb.remove(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_size_pre_s_69(HashSet sa, HashSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v1 ~: sa..contents" */
        boolean r1a = sa.remove(v1);
        int r2a = sa.size();

        int r2b = sb.size();
        boolean r1b = sb.remove(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void remove_size_pre_c_69(HashSet sa, HashSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 ~: sa..contents)" */
        boolean r1a = sa.remove(v1);
        int r2a = sa.size();

        int r2b = sb.size();
        boolean r1b = sb.remove(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void remove_size_between_s_70(HashSet sa, HashSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        /*: assume "~r1a" */
```

```
        int r2a = sa.size();

        int r2b = sb.size();
        boolean r1b = sb.remove(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void remove_size_between_c_70(HashSet sa, HashSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        /*: assume "~(~r1a)" */
        int r2a = sa.size();

        int r2b = sb.size();
        boolean r1b = sb.remove(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void remove_size_post_s_71(HashSet sa, HashSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        int r2a = sa.size();
        /*: assume "~r1a" */

        int r2b = sb.size();
        boolean r1b = sb.remove(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void remove_size_post_c_71(HashSet sa, HashSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.remove(v1);
        int r2a = sa.size();
        /*: assume "~(~r1a)" */

        int r2b = sb.size();
        boolean r1b = sb.remove(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void remove_add_pre_s_72(HashSet sa, HashSet sb, Object v1, Object v2)
```

```
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v1 ~= v2" */
        sa.remove(v1);
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        sb.remove(v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_add_pre_c_72(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 ~= v2)" */
        sa.remove(v1);
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        sb.remove(v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_add_between_s_73(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(v1);
        /*: assume "v1 ~= v2" */
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        sb.remove(v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_add_between_c_73(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(v1);
        /*: assume "~(v1 ~= v2)" */
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        sb.remove(v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
```

```
2356    }
2357
2358    static void remove_add_post_s_74(HashSet sa, HashSet sb, Object v1, Object v2)
2359    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
2360                sa..contents = sb..contents & sa..size = sb..size"
2361        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2362        ensures "True" */
2363    {
2364        sa.remove(v1);
2365        boolean r2a = sa.add(v2);
2366        /*: assume "v1 ~= v2" */
2367
2368        boolean r2b = sb.add(v2);
2369        sb.remove(v1);
2370
2371        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2372    }
2373
2374    static void remove_add_post_c_74(HashSet sa, HashSet sb, Object v1, Object v2)
2375    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
2376                sa..contents = sb..contents & sa..size = sb..size"
2377        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2378        ensures "True" */
2379    {
2380        sa.remove(v1);
2381        boolean r2a = sa.add(v2);
2382        /*: assume "~(v1 ~= v2)" */
2383
2384        boolean r2b = sb.add(v2);
2385        sb.remove(v1);
2386
2387        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2388    }
2389
2390    static void remove_add_pre_s_75(HashSet sa, HashSet sb, Object v1, Object v2)
2391    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
2392                sa..contents = sb..contents & sa..size = sb..size"
2393        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2394        ensures "True" */
2395    {
2396        /*: assume "v1 ~= v2" */
2397        sa.remove(v1);
2398        sa.add(v2);
2399
2400        sb.add(v2);
2401        sb.remove(v1);
2402
2403        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
2404    }
2405
2406    static void remove_add_pre_c_75(HashSet sa, HashSet sb, Object v1, Object v2)
2407    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            & v2 ~= null &
2408                sa..contents = sb..contents & sa..size = sb..size"
2409        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2410        ensures "True" */
2411    {
2412        /*: assume "~(v1 ~= v2)" */
2413        sa.remove(v1);
2414        sa.add(v2);
2415
2416        sb.add(v2);
```

```
2417        sb.remove(v1);
2418
2419        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
2420    }
2421
2422    static void remove_add_between_s_76(HashSet sa, HashSet sb, Object v1, Object v2)
2423    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
2424                sa..contents = sb..contents & sa..size = sb..size"
2425        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2426        ensures "True" */
2427    {
2428        sa.remove(v1);
2429        /*: assume "v1 ~= v2" */
2430        sa.add(v2);
2431
2432        sb.add(v2);
2433        sb.remove(v1);
2434
2435        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
2436    }
2437
2438    static void remove_add_between_c_76(HashSet sa, HashSet sb, Object v1, Object v2)
2439    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
2440                sa..contents = sb..contents & sa..size = sb..size"
2441        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2442        ensures "True" */
2443    {
2444        sa.remove(v1);
2445        /*: assume "~(v1 ~= v2)" */
2446        sa.add(v2);
2447
2448        sb.add(v2);
2449        sb.remove(v1);
2450
2451        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
2452    }
2453
2454    static void remove_add_post_s_77(HashSet sa, HashSet sb, Object v1, Object v2)
2455    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
2456                sa..contents = sb..contents & sa..size = sb..size"
2457        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2458        ensures "True" */
2459    {
2460        sa.remove(v1);
2461        sa.add(v2);
2462        /*: assume "v1 ~= v2" */
2463
2464        sb.add(v2);
2465        sb.remove(v1);
2466
2467        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
2468    }
2469
2470    static void remove_add_post_c_77(HashSet sa, HashSet sb, Object v1, Object v2)
2471    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
2472                sa..contents = sb..contents & sa..size = sb..size"
2473        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2474        ensures "True" */
2475    {
2476        sa.remove(v1);
2477        sa.add(v2);
```

```
        /*: assume "~(v1 ~= v2)" */

        sb.add(v2);
        sb.remove(v1);

        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_contains_pre_s_78(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v1 ~= v2 | v1 ~: sa..contents" */
        sa.remove(v1);
        boolean r2a = sa.contains(v2);

        boolean r2b = sb.contains(v2);
        sb.remove(v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_contains_pre_c_78(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 ~= v2 | v1 ~: sa..contents)" */
        sa.remove(v1);
        boolean r2a = sa.contains(v2);

        boolean r2b = sb.contains(v2);
        sb.remove(v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_contains_between_s_79(HashSet sa, HashSet sb, Object v1, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(v1);
        /*: assume "v1 ~= v2 | v1 ~: sa..(old contents)" */
        boolean r2a = sa.contains(v2);

        boolean r2b = sb.contains(v2);
        sb.remove(v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_contains_between_c_79(HashSet sa, HashSet sb, Object v1, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                   sa..contents = sb..contents & sa..size = sb..size"
```

```
2537        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2538        ensures "True" */
2539    {
2540        sa.remove(v1);
2541        /*: assume "~(v1 ~= v2 | v1 ~: sa..(old contents))" */
2542        boolean r2a = sa.contains(v2);
2543
2544        boolean r2b = sb.contains(v2);
2545        sb.remove(v1);
2546
2547        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2548    }
2549
2550    static void remove_contains_post_s_80(HashSet sa, HashSet sb, Object v1, Object v2)
2551    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
2552                sa..contents = sb..contents & sa..size = sb..size"
2553        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2554        ensures "True" */
2555    {
2556        sa.remove(v1);
2557        boolean r2a = sa.contains(v2);
2558        /*: assume "v1 ~= v2 | v1 ~: sa..(old contents)" */
2559
2560        boolean r2b = sb.contains(v2);
2561        sb.remove(v1);
2562
2563        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2564    }
2565
2566    static void remove_contains_post_c_80(HashSet sa, HashSet sb, Object v1, Object v2)
2567    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
2568                sa..contents = sb..contents & sa..size = sb..size"
2569        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2570        ensures "True" */
2571    {
2572        sa.remove(v1);
2573        boolean r2a = sa.contains(v2);
2574        /*: assume "~(v1 ~= v2 | v1 ~: sa..(old contents))" */
2575
2576        boolean r2b = sb.contains(v2);
2577        sb.remove(v1);
2578
2579        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2580    }
2581
2582    static void remove_remove_pre_s_81(HashSet sa, HashSet sb, Object v1, Object v2)
2583    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
2584                sa..contents = sb..contents & sa..size = sb..size"
2585        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2586        ensures "True" */
2587    {
2588        /*: assume "v1 ~= v2 | v1 ~: sa..contents" */
2589        sa.remove(v1);
2590        boolean r2a = sa.remove(v2);
2591
2592        boolean r2b = sb.remove(v2);
2593        sb.remove(v1);
2594
2595        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2596    }
2597
2598    static void remove_remove_pre_c_81(HashSet sa, HashSet sb, Object v1, Object v2)
```

```
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v1 ~= v2 | v1 ~: sa..contents)" */
        sa.remove(v1);
        boolean r2a = sa.remove(v2);

        boolean r2b = sb.remove(v2);
        sb.remove(v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_remove_between_s_82(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(v1);
        /*: assume "v1 ~= v2 | v1 ~: sa..(old contents)" */
        boolean r2a = sa.remove(v2);

        boolean r2b = sb.remove(v2);
        sb.remove(v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_remove_between_c_82(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(v1);
        /*: assume "~(v1 ~= v2 | v1 ~: sa..(old contents))" */
        boolean r2a = sa.remove(v2);

        boolean r2b = sb.remove(v2);
        sb.remove(v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_remove_post_s_83(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(v1);
        boolean r2a = sa.remove(v2);
        /*: assume "v1 ~= v2 | v1 ~: sa..(old contents)" */

        boolean r2b = sb.remove(v2);
        sb.remove(v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
```

```
2660     }
2661
2662     static void remove_remove_post_c_83(HashSet sa, HashSet sb, Object v1, Object v2)
2663     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
             & v2 ~= null &
2664                 sa..contents = sb..contents & sa..size = sb..size"
2665          modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2666          ensures "True" */
2667     {
2668         sa.remove(v1);
2669         boolean r2a = sa.remove(v2);
2670         /*: assume "~(v1 ~= v2 | v1 ~: sa..(old contents))" */
2671
2672         boolean r2b = sb.remove(v2);
2673         sb.remove(v1);
2674
2675         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2676     }
2677
2678     static void remove_remove_pre_s_84(HashSet sa, HashSet sb, Object v1, Object v2)
2679     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
             & v2 ~= null &
2680                 sa..contents = sb..contents & sa..size = sb..size"
2681          modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2682          ensures "True" */
2683     {
2684         /*: assume "True" */
2685         sa.remove(v1);
2686         sa.remove(v2);
2687
2688         sb.remove(v2);
2689         sb.remove(v1);
2690
2691         /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
2692     }
2693
2694     static void remove_remove_pre_c_84(HashSet sa, HashSet sb, Object v1, Object v2)
2695     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
             & v2 ~= null &
2696                 sa..contents = sb..contents & sa..size = sb..size"
2697          modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2698          ensures "True" */
2699     {
2700         /*: assume "~(True)" */
2701         sa.remove(v1);
2702         sa.remove(v2);
2703
2704         sb.remove(v2);
2705         sb.remove(v1);
2706
2707         /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
2708     }
2709
2710     static void remove_remove_between_s_85(HashSet sa, HashSet sb, Object v1, Object v2)
2711     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
             & v2 ~= null &
2712                 sa..contents = sb..contents & sa..size = sb..size"
2713          modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2714          ensures "True" */
2715     {
2716         sa.remove(v1);
2717         /*: assume "True" */
2718         sa.remove(v2);
2719
2720         sb.remove(v2);
```

```
        sb.remove(v1);

        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_remove_between_c_85(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(v1);
        /*: assume "~(True)" */
        sa.remove(v2);

        sb.remove(v2);
        sb.remove(v1);

        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_remove_post_s_86(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(v1);
        sa.remove(v2);
        /*: assume "True" */

        sb.remove(v2);
        sb.remove(v1);

        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_remove_post_c_86(HashSet sa, HashSet sb, Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(v1);
        sa.remove(v2);
        /*: assume "~(True)" */

        sb.remove(v2);
        sb.remove(v1);

        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_size_pre_s_87(HashSet sa, HashSet sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v1 ~: sa..contents" */
        sa.remove(v1);
```

```
2782        int r2a = sa.size();
2783
2784        int r2b = sb.size();
2785        sb.remove(v1);
2786
2787        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2788    }
2789
2790    static void remove_size_pre_c_87(HashSet sa, HashSet sb, Object v1)
2791    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        &
2792                sa..contents = sb..contents & sa..size = sb..size"
2793        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2794        ensures "True" */
2795    {
2796        /*: assume "~(v1 ~: sa..contents)" */
2797        sa.remove(v1);
2798        int r2a = sa.size();
2799
2800        int r2b = sb.size();
2801        sb.remove(v1);
2802
2803        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2804    }
2805
2806    static void remove_size_between_s_88(HashSet sa, HashSet sb, Object v1)
2807    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        &
2808                sa..contents = sb..contents & sa..size = sb..size"
2809        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2810        ensures "True" */
2811    {
2812        sa.remove(v1);
2813        /*: assume "v1 ~: sa..(old contents)" */
2814        int r2a = sa.size();
2815
2816        int r2b = sb.size();
2817        sb.remove(v1);
2818
2819        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2820    }
2821
2822    static void remove_size_between_c_88(HashSet sa, HashSet sb, Object v1)
2823    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        &
2824                sa..contents = sb..contents & sa..size = sb..size"
2825        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2826        ensures "True" */
2827    {
2828        sa.remove(v1);
2829        /*: assume "~(v1 ~: sa..(old contents))" */
2830        int r2a = sa.size();
2831
2832        int r2b = sb.size();
2833        sb.remove(v1);
2834
2835        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2836    }
2837
2838    static void remove_size_post_s_89(HashSet sa, HashSet sb, Object v1)
2839    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        &
2840                sa..contents = sb..contents & sa..size = sb..size"
2841        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2842        ensures "True" */
```

```
2843    {
2844        sa.remove(v1);
2845        int r2a = sa.size();
2846        /*: assume "v1 ~: sa..(old contents)" */
2847
2848        int r2b = sb.size();
2849        sb.remove(v1);
2850
2851        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2852    }
2853
2854    static void remove_size_post_c_89(HashSet sa, HashSet sb, Object v1)
2855    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
            &
2856                   sa..contents = sb..contents & sa..size = sb..size"
2857        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2858        ensures "True" */
2859    {
2860        sa.remove(v1);
2861        int r2a = sa.size();
2862        /*: assume "~(v1 ~: sa..(old contents))" */
2863
2864        int r2b = sb.size();
2865        sb.remove(v1);
2866
2867        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2868    }
2869
2870    static void size_add_pre_s_90(HashSet sa, HashSet sb, Object v2)
2871    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
            &
2872                   sa..contents = sb..contents & sa..size = sb..size"
2873        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2874        ensures "True" */
2875    {
2876        /*: assume "v2 : sa..contents" */
2877        int r1a = sa.size();
2878        boolean r2a = sa.add(v2);
2879
2880        boolean r2b = sb.add(v2);
2881        int r1b = sb.size();
2882
2883        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
2884    }
2885
2886    static void size_add_pre_c_90(HashSet sa, HashSet sb, Object v2)
2887    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
            &
2888                   sa..contents = sb..contents & sa..size = sb..size"
2889        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2890        ensures "True" */
2891    {
2892        /*: assume "~(v2 : sa..contents)" */
2893        int r1a = sa.size();
2894        boolean r2a = sa.add(v2);
2895
2896        boolean r2b = sb.add(v2);
2897        int r1b = sb.size();
2898
2899        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
2900    }
2901
2902    static void size_add_between_s_91(HashSet sa, HashSet sb, Object v2)
```

```
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
        &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "v2 : sa..contents" */
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void size_add_between_c_91(HashSet sa, HashSet sb, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
        &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "~(v2 : sa..contents)" */
        boolean r2a = sa.add(v2);

        boolean r2b = sb.add(v2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void size_add_post_s_92(HashSet sa, HashSet sb, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
        &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
        boolean r2a = sa.add(v2);
        /*: assume "~r2a" */

        boolean r2b = sb.add(v2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void size_add_post_c_92(HashSet sa, HashSet sb, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
        &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
        boolean r2a = sa.add(v2);
        /*: assume "~(~r2a)" */

        boolean r2b = sb.add(v2);
```

```
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void size_add_pre_s_93(HashSet sa, HashSet sb, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "v2 : sa..contents" */
        int r1a = sa.size();
        sa.add(v2);

        sb.add(v2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void size_add_pre_c_93(HashSet sa, HashSet sb, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(v2 : sa..contents)" */
        int r1a = sa.size();
        sa.add(v2);

        sb.add(v2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void size_add_between_s_94(HashSet sa, HashSet sb, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "v2 : sa..contents" */
        sa.add(v2);

        sb.add(v2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void size_add_between_c_94(HashSet sa, HashSet sb, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
```

```
3021        /*: assume "~(v2 : sa..contents)" */
3022        sa.add(v2);
3023
3024        sb.add(v2);
3025        int r1b = sb.size();
3026
3027        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
3028    }
3029
3030    static void size_add_post_s_95(HashSet sa, HashSet sb, Object v2)
3031    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
        &
3032                   sa..contents = sb..contents & sa..size = sb..size"
3033        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3034        ensures "True" */
3035    {
3036        int r1a = sa.size();
3037        /*: ghost specvar sa__contents :: "obj set" = "sa..contents" */
3038        sa.add(v2);
3039        /*: assume "v2 : sa__contents" */
3040
3041        sb.add(v2);
3042        int r1b = sb.size();
3043
3044        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
3045    }
3046
3047    static void size_add_post_c_95(HashSet sa, HashSet sb, Object v2)
3048    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
        &
3049                   sa..contents = sb..contents & sa..size = sb..size"
3050        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3051        ensures "True" */
3052    {
3053        int r1a = sa.size();
3054        /*: ghost specvar sa__contents :: "obj set" = "sa..contents" */
3055        sa.add(v2);
3056        /*: assume "~(v2 : sa__contents)" */
3057
3058        sb.add(v2);
3059        int r1b = sb.size();
3060
3061        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
3062    }
3063
3064    static void size_contains_pre_s_96(HashSet sa, HashSet sb, Object v2)
3065    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
        &
3066                   sa..contents = sb..contents & sa..size = sb..size"
3067        ensures "True" */
3068    {
3069        /*: assume "True" */
3070        int r1a = sa.size();
3071        boolean r2a = sa.contains(v2);
3072
3073        boolean r2b = sb.contains(v2);
3074        int r1b = sb.size();
3075
3076        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
3077    }
3078
3079    static void size_contains_pre_c_96(HashSet sa, HashSet sb, Object v2)
3080    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
        &
```

```
3081                 sa..contents = sb..contents & sa..size = sb..size"
3082         ensures "True" */
3083     {
3084         /*: assume "~(True)" */
3085         int r1a = sa.size();
3086         boolean r2a = sa.contains(v2);
3087 
3088         boolean r2b = sb.contains(v2);
3089         int r1b = sb.size();
3090 
3091         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
3092     }
3093 
3094     static void size_contains_between_s_97(HashSet sa, HashSet sb, Object v2)
3095     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
            &
3096                 sa..contents = sb..contents & sa..size = sb..size"
3097         ensures "True" */
3098     {
3099         int r1a = sa.size();
3100         /*: assume "True" */
3101         boolean r2a = sa.contains(v2);
3102 
3103         boolean r2b = sb.contains(v2);
3104         int r1b = sb.size();
3105 
3106         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
3107     }
3108 
3109     static void size_contains_between_c_97(HashSet sa, HashSet sb, Object v2)
3110     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
            &
3111                 sa..contents = sb..contents & sa..size = sb..size"
3112         ensures "True" */
3113     {
3114         int r1a = sa.size();
3115         /*: assume "~(True)" */
3116         boolean r2a = sa.contains(v2);
3117 
3118         boolean r2b = sb.contains(v2);
3119         int r1b = sb.size();
3120 
3121         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
3122     }
3123 
3124     static void size_contains_post_s_98(HashSet sa, HashSet sb, Object v2)
3125     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
            &
3126                 sa..contents = sb..contents & sa..size = sb..size"
3127         ensures "True" */
3128     {
3129         int r1a = sa.size();
3130         boolean r2a = sa.contains(v2);
3131         /*: assume "True" */
3132 
3133         boolean r2b = sb.contains(v2);
3134         int r1b = sb.size();
3135 
3136         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
3137     }
3138 
```

```
3139    static void size_contains_post_c_98(HashSet sa, HashSet sb, Object v2)
3140    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
            &
3141                sa..contents = sb..contents & sa..size = sb..size"
3142        ensures "True" */
3143    {
3144        int r1a = sa.size();
3145        boolean r2a = sa.contains(v2);
3146        /*: assume "~(True)" */
3147
3148        boolean r2b = sb.contains(v2);
3149        int r1b = sb.size();
3150
3151        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
3152    }
3153
3154    static void size_remove_pre_s_99(HashSet sa, HashSet sb, Object v2)
3155    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
            &
3156                sa..contents = sb..contents & sa..size = sb..size"
3157        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3158        ensures "True" */
3159    {
3160        /*: assume "v2 ~: sa..contents" */
3161        int r1a = sa.size();
3162        boolean r2a = sa.remove(v2);
3163
3164        boolean r2b = sb.remove(v2);
3165        int r1b = sb.size();
3166
3167        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
3168    }
3169
3170    static void size_remove_pre_c_99(HashSet sa, HashSet sb, Object v2)
3171    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
            &
3172                sa..contents = sb..contents & sa..size = sb..size"
3173        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3174        ensures "True" */
3175    {
3176        /*: assume "~(v2 ~: sa..contents)" */
3177        int r1a = sa.size();
3178        boolean r2a = sa.remove(v2);
3179
3180        boolean r2b = sb.remove(v2);
3181        int r1b = sb.size();
3182
3183        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
3184    }
3185
3186    static void size_remove_between_s_100(HashSet sa, HashSet sb, Object v2)
3187    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
            &
3188                sa..contents = sb..contents & sa..size = sb..size"
3189        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3190        ensures "True" */
3191    {
3192        int r1a = sa.size();
3193        /*: assume "v2 ~: sa..contents" */
3194        boolean r2a = sa.remove(v2);
3195
3196        boolean r2b = sb.remove(v2);
```

```
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void size_remove_between_c_100(HashSet sa, HashSet sb, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "~(v2 ~: sa..contents)" */
        boolean r2a = sa.remove(v2);

        boolean r2b = sb.remove(v2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void size_remove_post_s_101(HashSet sa, HashSet sb, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
        boolean r2a = sa.remove(v2);
        /*: assume "~r2a" */

        boolean r2b = sb.remove(v2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void size_remove_post_c_101(HashSet sa, HashSet sb, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
        boolean r2a = sa.remove(v2);
        /*: assume "~(~r2a)" */

        boolean r2b = sb.remove(v2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void size_remove_pre_s_102(HashSet sa, HashSet sb, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
```

```
3254            ensures "True" */
3255        {
3256            /*: assume "v2 ~: sa..contents" */
3257            int r1a = sa.size();
3258            sa.remove(v2);
3259
3260            sb.remove(v2);
3261            int r1b = sb.size();
3262
3263            /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
3264        }
3265
3266        static void size_remove_pre_c_102(HashSet sa, HashSet sb, Object v2)
3267        /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
                &
3268                        sa..contents = sb..contents & sa..size = sb..size"
3269            modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3270            ensures "True" */
3271        {
3272            /*: assume "~(v2 ~: sa..contents)" */
3273            int r1a = sa.size();
3274            sa.remove(v2);
3275
3276            sb.remove(v2);
3277            int r1b = sb.size();
3278
3279            /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
3280        }
3281
3282        static void size_remove_between_s_103(HashSet sa, HashSet sb, Object v2)
3283        /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
                &
3284                        sa..contents = sb..contents & sa..size = sb..size"
3285            modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3286            ensures "True" */
3287        {
3288            int r1a = sa.size();
3289            /*: assume "v2 ~: sa..contents" */
3290            sa.remove(v2);
3291
3292            sb.remove(v2);
3293            int r1b = sb.size();
3294
3295            /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
3296        }
3297
3298        static void size_remove_between_c_103(HashSet sa, HashSet sb, Object v2)
3299        /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
                &
3300                        sa..contents = sb..contents & sa..size = sb..size"
3301            modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3302            ensures "True" */
3303        {
3304            int r1a = sa.size();
3305            /*: assume "~(v2 ~: sa..contents)" */
3306            sa.remove(v2);
3307
3308            sb.remove(v2);
3309            int r1b = sb.size();
3310
3311            /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
3312        }
3313
3314        static void size_remove_post_s_104(HashSet sa, HashSet sb, Object v2)
```

```
3315    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
            &
3316                sa..contents = sb..contents & sa..size = sb..size"
3317        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3318        ensures "True" */
3319    {
3320        int r1a = sa.size();
3321        /*: ghost specvar sa__contents :: "obj set" = "sa..contents" */
3322        sa.remove(v2);
3323        /*: assume "v2 ~: sa__contents" */
3324
3325        sb.remove(v2);
3326        int r1b = sb.size();
3327
3328        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
3329    }
3330
3331    static void size_remove_post_c_104(HashSet sa, HashSet sb, Object v2)
3332    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
            &
3333                sa..contents = sb..contents & sa..size = sb..size"
3334        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3335        ensures "True" */
3336    {
3337        int r1a = sa.size();
3338        /*: ghost specvar sa__contents :: "obj set" = "sa..contents" */
3339        sa.remove(v2);
3340        /*: assume "~(v2 ~: sa__contents)" */
3341
3342        sb.remove(v2);
3343        int r1b = sb.size();
3344
3345        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
3346    }
3347
3348    static void size_size_pre_s_105(HashSet sa, HashSet sb)
3349    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3350                sa..contents = sb..contents & sa..size = sb..size"
3351        ensures "True" */
3352    {
3353        /*: assume "True" */
3354        int r1a = sa.size();
3355        int r2a = sa.size();
3356
3357        int r2b = sb.size();
3358        int r1b = sb.size();
3359
3360        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
3361    }
3362
3363    static void size_size_pre_c_105(HashSet sa, HashSet sb)
3364    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3365                sa..contents = sb..contents & sa..size = sb..size"
3366        ensures "True" */
3367    {
3368        /*: assume "~(True)" */
3369        int r1a = sa.size();
3370        int r2a = sa.size();
3371
3372        int r2b = sb.size();
3373        int r1b = sb.size();
3374
3375        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
```

```
3376    }
3377
3378    static void size_size_between_s_106(HashSet sa, HashSet sb)
3379    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3380                sa..contents = sb..contents & sa..size = sb..size"
3381        ensures "True" */
3382    {
3383        int r1a = sa.size();
3384        /*: assume "True" */
3385        int r2a = sa.size();
3386
3387        int r2b = sb.size();
3388        int r1b = sb.size();
3389
3390        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
3391    }
3392
3393    static void size_size_between_c_106(HashSet sa, HashSet sb)
3394    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3395                sa..contents = sb..contents & sa..size = sb..size"
3396        ensures "True" */
3397    {
3398        int r1a = sa.size();
3399        /*: assume "~(True)" */
3400        int r2a = sa.size();
3401
3402        int r2b = sb.size();
3403        int r1b = sb.size();
3404
3405        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
3406    }
3407
3408    static void size_size_post_s_107(HashSet sa, HashSet sb)
3409    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3410                sa..contents = sb..contents & sa..size = sb..size"
3411        ensures "True" */
3412    {
3413        int r1a = sa.size();
3414        int r2a = sa.size();
3415        /*: assume "True" */
3416
3417        int r2b = sb.size();
3418        int r1b = sb.size();
3419
3420        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
3421    }
3422
3423    static void size_size_post_c_107(HashSet sa, HashSet sb)
3424    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3425                sa..contents = sb..contents & sa..size = sb..size"
3426        ensures "True" */
3427    {
3428        int r1a = sa.size();
3429        int r2a = sa.size();
3430        /*: assume "~(True)" */
3431
3432        int r2b = sb.size();
3433        int r1b = sb.size();
3434
3435        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
3436    }
```

```
}
```

B.3.3 Inverse Testing Methods

Listing 9. HashSetInv.java

```
class HashSetInv {
    static void add_0(HashSet s, Object v)
    /*: requires "s ~= null & s..init & v ~= null"
        modifies "s..contents", "s..size"
        ensures "True" */
    {
        boolean r = s.add(v);
        if (r) { s.remove(v); }

        /*: assert "s..contents = s..(old contents) & s..size = s..(old size)" */
    }

    static void remove_1(HashSet s, Object v)
    /*: requires "s ~= null & s..init & v ~= null"
        modifies "s..contents", "s..size"
        ensures "True" */
    {
        boolean r = s.remove(v);
        if (r) { s.add(v); }

        /*: assert "s..contents = s..(old contents) & s..size = s..(old size)" */
    }

}
```

B.4 AssociationList

B.4.1 Data Structure

Listing 10. AssociationList.java

```
public /*: claimedby AssociationList */ class Node {
    public Object key;
    public Object value;
    public Node next;

    /*: public ghost specvar conts :: "(obj * obj) set" = "{}"
        invariant CntDef: "ALL x. x : Node & x : alloc & x ~= null --> x..conts =
            {(x..key, x..value)} Un x..next..conts & (ALL v. (x..key, v) ~:
            x..next..conts)"
        invariant CntNull: "null..conts = {}" */
}

public class AssociationList {
    private Node head;
    private int _size;

    /*: public specvar contents :: "(obj * obj) set"
        vardefs "contents == head..conts"
        public ensured invariant MapInv: "ALL k v0 v1. (k, v0) : contents & (k, v1) :
            contents --> v0 = v1"
        invariant NonNullInv: "ALL k v. (k, v) : contents --> k ~= null & v ~= null"

        static specvar edge :: "obj => obj => bool"
        vardefs "edge == (%x y. (x : Node & y = x..next) | (x : AssociationList & y =
            x..head))"
        invariant InjInv: "ALL x1 x2 y. y ~= null & edge x1 y & edge x2 y --> x1 = x2"

        public specvar size :: "int"
```

```
25          vardefs "size == _size"
26          invariant CardInv: "_size = card (contents)" */
27
28      public AssociationList()
29      /*: modifies "contents", "size"
30          ensures "contents = {} & size = 0" */
31      {
32          head = null;
33          _size = 0;
34
35          {
36              /*: pickAny x :: obj */
37              {
38                  /*: assuming CardHyp: "x : alloc & x : AssociationList" */
39                  {
40                      /*: assuming XIsThisHyp: "x = this" */
41                      /*: note LengthZero: "x.._size = 0" */
42                      /*: note ContentsEmpty: "x..contents = {}" */
43                      /*: note XIsThisCard: "x.._size = card (x..contents)" from
                            XIsThisHyp, LengthZero, ContentsEmpty */
44                  }
45                  {
46                      /*: assuming XNotThisHyp: "x ~= this" */
47                      /*: note XInOldAlloc: "x : old alloc" */
48                      /*: note OldCard: "x..(old AssociationList._size) = card (x..(old
                            AssociationList.contents))" from CardHyp, XNotThisHyp,
                            XInOldAlloc, CardInv */
49                      /*: note XLengthEq: "x.._size = x..(old AssociationList._size)" */
50                      /*: note XContentsUnchanged: "x..contents = x..(old
                            AssociationList.contents)" */
51                      /*: note XNotThisCard: "x.._size = card (x..contents)" from
                            XLengthEq, OldCard, XContentsUnchanged */
52                  }
53                  /*: note CardConc: "x.._size = card (x..contents)" from XIsThisCard,
                        XNotThisCard */
54              }
55              /*: note CardPostCond: "x : alloc & x : AssociationList --> x.._size = card
                    (x..contents)" forSuch x */
56          }
57      }
58
59      public boolean containsKey(Object k0)
60      /*: ensures "result = (EX v. ((k0, v) : contents))" */
61      {
62          return _containsKey(k0);
63      }
64
65      private boolean _containsKey(Object k0)
66      /*: requires "theinvs"
67          ensures "result = (EX v. ((k0, v) : contents)) & theinvs" */
68      {
69          Node curr = head;
70          while /*: invariant "(EX v. (k0, v) : contents) = (EX v. (k0, v) :
                curr..conts)" */ (curr != null) {
71              if (curr.key == k0) {
72                  return true;
73              }
74              curr = curr.next;
75          }
76          return false;
77      }
78
79      private void _add(Object k0, Object v0)
80      /*: requires "k0 ~= null & v0 ~= null & ~(EX v. (k0, v) : contents) & theinvs"
```

```
81          modifies "contents", "size", "new..key", "new..value", "new..next",
                "new..conts", "head", "_size"
82          ensures "contents = old contents Un {(k0, v0)} & size = old size + 1 & theinvs"
                */
83      {
84          Node n = new Node();
85          n.key = k0;
86          n.value = v0;
87          n.next = head;
88          /*: "n..conts" := "{(k0, v0)} Un head..conts" */
89          head = n;
90
91          _size = _size + 1;
92
93          /*: note ContentsPost: "contents = old contents Un {(k0, v0)}" */
94          {
95              /*: pickAny x :: obj */
96              {
97                  /*: assuming CardHyp: "x : alloc & x : AssociationList" */
98                  {
99                      /*: note ThisProps: "this : old alloc & this : AssociationList" */
100                     /*: note OldCard: "old _size = card (old contents)" from ThisProps,
                            CardInv */
101                     /*: note NewLength: "_size = old _size + 1" */
102                     /*: note NewNotInOld: "(k0, v0) ~: old contents" */
103                     /*: note XIsThisCard: "_size = card (contents)" from OldCard,
                            NewLength, NewNotInOld, ContentsPost */
104                 }
105                 {
106                     /*: assuming XNotThisHyp: "x ~= this" */
107                     /*: note XInOldAlloc: "x : old alloc" */
108                     /*: note OldCard: "x..(old AssociationList._size) = card (x..(old
                            AssociationList.contents))" from CardHyp, XNotThisHyp,
                            XInOldAlloc, CardInv */
109                     /*: note XLengthEq: "x.._size = x..(old AssociationList._size)" */
110                     {
111                         /*: localize */
112                         /*: note XContentsForw: "ALL y z. (y, z) : x..contents --> (y,
                                z) : x..(old AssociationList.contents)" */
113                         /*: note XContentsBack: "ALL y z. (y, z) : x..(old
                                AssociationList.contents) --> (y, z) : x..contents" */
114                         /*: note XContentsUnchanged: "x..contents = x..(old
                                AssociationList.contents)" from XContentsForw, XContentsBack
                                */
115                     }
116                     /*: note XNotThisCard: "x.._size = card (x..contents)" from
                            XLengthEq, OldCard, XContentsUnchanged */
117                 }
118                 /*: note CardConc: "x.._size = card (x..contents)" from XIsThisCard,
                        XNotThisCard */
119             }
120             /*: note CardPostCond: "x : alloc & x : AssociationList --> x.._size = card
                    (x..contents)" forSuch x */
121         }
122     }
123
124     public Object put(Object k0, Object v0)
125     /*: requires "k0 ~= null & v0 ~= null"
126         modifies "contents", "size"
127         ensures "((EX v. (k0, v) : old contents) --> (k0, result) : old contents &
                contents = old contents - {(k0, result)} Un {(k0, v0)} & size = old size &
                result ~= null) & (~(EX v. (k0, v) : old contents) --> contents = old
                contents Un {(k0, v0)} & size = old size + 1 & result = null) & (result ~=
                null --> (k0, result) : old contents & old contents = contents - {(k0, v0)}
                Un {(k0, result)} & old size = size) & (result = null --> ~(EX v. (k0, v) :
```

```
        old contents) & old contents = contents - {(k0, v0)} & old size = size - 1)"
        */
128 {
129     if (_containsKey(k0)) {
130         Object v1 = _remove(k0);
131         _add(k0, v0);
132         return v1;
133     } else {
134         _add(k0, v0);
135         return null;
136     }
137 }
138
139 public Object get(Object k0)
140 /*: requires "k0 ~= null"
141     ensures "((EX v. (k0, v) : contents) --> (k0, result) : contents & result ~=
        null) & (~(EX v. (k0, v) : contents) --> result = null)" */
142 {
143     Node curr = head;
144     while /*: invariant "ALL v. ((k0, v) : contents) = ((k0, v) : curr..conts)" */
        (curr != null) {
145         if (curr.key == k0) {
146             return curr.value;
147         }
148         curr = curr.next;
149     }
150     return null;
151 }
152
153 public Object remove(Object k0)
154 /*: requires "k0 ~= null"
155     modifies "contents", "size"
156     ensures "((EX v. (k0, v) : old contents) -->  (k0, result) : old contents &
        contents = old contents - {(k0, result)} & size = old size - 1 & result ~=
        null) & (~(EX v. (k0, v) : old contents) --> contents = old contents & size
        = old size & result = null)" */
157 {
158     if (_containsKey(k0))
159         return _remove(k0);
160     else
161         return null;
162 }
163
164 private Object _remove(Object k0)
165 /*: requires "k0 ~= null & (EX v. (k0, v) : contents) & theinvs"
166     modifies "contents", "size", "head", "next", "conts", "_size"
167     ensures "(k0, result) : old contents & contents = old contents - {(k0, result)}
        & size = old size - 1 & theinvs" */
168 {
169     /*: ghost specvar v0 :: obj */
170     /*: havoc v0 suchThat "(k0, v0) : contents" */
171     Node f = head;
172     if (f.key == k0) {
173         Node second = f.next;
174         f.next = null;
175         /*: "f..conts" := "{(f..key, f..value)}" */
176         head = second;
177         _size = _size - 1;
178         /*: note ContentsPost: "contents = old contents - {(f..key, f..value)}" */
179         {
180             /*: pickAny x :: obj */
181             {
182                 /*: assuming CardHyp: "x : alloc & x : AssociationList" */
183                 {
```

```
                    /*: note ThisProps: "this : old alloc & this : AssociationList"
                        */
                    /*: note OldCard: "old _size = card (old contents)" from
                        ThisProps, CardInv */
                    /*: note NewLength: "_size = old _size - 1" */
                    /*: note NewNotInOld: "(f..key, f..value) : old contents" */
                    /*: note XIsThisCard: "_size = card (contents)" from OldCard,
                        NewLength, NewNotInOld, ContentsPost */
                }
                {
                    /*: assuming XNotThisHyp: "x ~= this" */
                    /*: note XInOldAlloc: "x : old alloc" */
                    /*: note OldCard: "x..(old AssociationList._size) = card
                        (x..(old AssociationList.contents))" from CardHyp,
                        XNotThisHyp, XInOldAlloc, CardInv */
                    /*: note XLengthEq: "x.._size = x..(old AssociationList._size)"
                        */
                    {
                        /*: localize */
                        /*: note XContentsForw: "ALL y z. (y, z) : x..contents -->
                            (y, z) : x..(old AssociationList.contents)" */
                        /*: note XContentsBack: "ALL y z. (y, z) : x..(old
                            AssociationList.contents) --> (y, z) : x..contents" */
                        /*: note XContentsUnchanged: "x..contents = x..(old
                            AssociationList.contents)" from XContentsForw,
                            XContentsBack */
                    }
                    /*: note XNotThisCard: "x.._size = card (x..contents)" from
                        XLengthEq, OldCard, XContentsUnchanged */
                }
                /*: note CardConc: "x.._size = card (x..contents)" from
                    XIsThisCard, XNotThisCard */
            }
            /*: note CardPostCond: "x : alloc & x : AssociationList --> x.._size =
                card (x..contents)" forSuch x */
        }
        return f.value;
    } else {
        Node prev = head;
        /*: "prev..conts" := "prev..conts - {(k0, v0)}" */
        Node curr = prev.next;
        while /*: invariant "prev ~= null & prev..conts = prev..(old conts) - {(k0,
            v0)} & curr ~= null & prev..next = curr & prev ~= curr & contents = old
            contents - {(k0, v0)} & (ALL n. n : AssociationList & n : old alloc & n
            ~= this --> n..contents = old (n..contents)) & (k0, v0) : curr..conts &
            comment ''CntDefInv'' (ALL n. n : Node & n : alloc & n ~= null & n ~=
            prev --> n..conts = {(n..key, n..value)} Un n..next..conts & (ALL v.
            (n..key, v) ~: n..next..conts)) & (ALL n. n..conts = old (n..conts) |
            n..conts = old (n..conts) - {(k0, v0)}) & null..conts = {}" */ (curr.key
            != k0) {
            /*: "curr..conts" := "curr..conts - {(k0, v0)}" */
            prev = curr;
            curr = curr.next;
        }
        Node tmp = curr.next;
        prev.next = tmp;
        curr.next = null;
        /*: "curr..conts" := "{(curr..key, curr..value)}" */
        _size = _size - 1;
        /*: note ContentsPost: "contents = old contents - {(curr..key,
            curr..value)}" */
        {
            /*: pickAny x :: obj */
            {
                /*: assuming CardHyp: "x : alloc & x : AssociationList" */
```

```
                    {
                        /*: note ThisProps: "this : old alloc & this : AssociationList"
                            */
                        /*: note OldCard: "old _size = card (old contents)" from
                            ThisProps, CardInv */
                        /*: note NewLength: "_size = old _size - 1" */
                        /*: note NewNotInOld: "(curr..key, curr..value) : old contents"
                            */
                        /*: note XIsThisCard: "_size = card (contents)" from OldCard,
                            NewLength, NewNotInOld, ContentsPost */
                    }
                    {
                        /*: assuming XNotThisHyp: "x ~= this" */
                        /*: note XInOldAlloc: "x : old alloc" */
                        /*: note OldCard: "x..(old AssociationList._size) = card
                            (x..(old AssociationList.contents))" from CardHyp,
                            XNotThisHyp, XInOldAlloc, CardInv */
                        /*: note XLengthEq: "x.._size = x..(old AssociationList._size)"
                            */
                        {
                            /*: localize */
                            /*: note XContentsForw: "ALL y z. (y, z) : x..contents -->
                                (y, z) : x..(old AssociationList.contents)" */
                            /*: note XContentsBack: "ALL y z. (y, z) : x..(old
                                AssociationList.contents) --> (y, z) : x..contents" */
                            /*: note XContentsUnchanged: "x..contents = x..(old
                                AssociationList.contents)" from XContentsForw,
                                XContentsBack */
                        }
                        /*: note XNotThisCard: "x.._size = card (x..contents)" from
                            XLengthEq, OldCard, XContentsUnchanged */
                    }
                    /*: note CardConc: "x.._size = card (x..contents)" from
                        XIsThisCard, XNotThisCard */
                }
                /*: note CardPostCond: "x : alloc & x : AssociationList --> x.._size =
                    card (x..contents)" forSuch x */
            }
            return curr.value;
        }
    }

    public int size()
    /*: ensures "result = size" */
    {
        return _size;
    }
}
```

B.4.2 Commutativity Testing Methods

Listing 11. AssociationListComm.java

```
class AssociationListComm {
    static void containsKey_containsKey_pre_s_0(AssociationList sa, AssociationList sb,
        Object k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
                   sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        /*: assume "True" */
        boolean r1a = sa.containsKey(k1);
        boolean r2a = sa.containsKey(k2);

        boolean r2b = sb.containsKey(k2);
```

```
        boolean r1b = sb.containsKey(k1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void containsKey_containsKey_pre_c_0(AssociationList sa, AssociationList sb,
        Object k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        /*: assume "~(True)" */
        boolean r1a = sa.containsKey(k1);
        boolean r2a = sa.containsKey(k2);

        boolean r2b = sb.containsKey(k2);
        boolean r1b = sb.containsKey(k1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void containsKey_containsKey_between_s_1(AssociationList sa, AssociationList
        sb, Object k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        boolean r1a = sa.containsKey(k1);
        /*: assume "True" */
        boolean r2a = sa.containsKey(k2);

        boolean r2b = sb.containsKey(k2);
        boolean r1b = sb.containsKey(k1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void containsKey_containsKey_between_c_1(AssociationList sa, AssociationList
        sb, Object k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        boolean r1a = sa.containsKey(k1);
        /*: assume "~(True)" */
        boolean r2a = sa.containsKey(k2);

        boolean r2b = sb.containsKey(k2);
        boolean r1b = sb.containsKey(k1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void containsKey_containsKey_post_s_2(AssociationList sa, AssociationList
        sb, Object k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        boolean r1a = sa.containsKey(k1);
        boolean r2a = sa.containsKey(k2);
```

```
69          /*: assume "True" */
70
71          boolean r2b = sb.containsKey(k2);
72          boolean r1b = sb.containsKey(k1);
73
74          /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
75      }
76
77      static void containsKey_containsKey_post_c_2(AssociationList sa, AssociationList
            sb, Object k1, Object k2)
78      /*: requires "sa ~= null & sb ~= null & sa ~= sb &
79                  sa..contents = sb..contents & sa..size = sb..size"
80          ensures "True" */
81      {
82          boolean r1a = sa.containsKey(k1);
83          boolean r2a = sa.containsKey(k2);
84          /*: assume "~(True)" */
85
86          boolean r2b = sb.containsKey(k2);
87          boolean r1b = sb.containsKey(k1);
88
89          /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
90      }
91
92      static void containsKey_get_pre_s_3(AssociationList sa, AssociationList sb, Object
            k1, Object k2)
93      /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
94                  sa..contents = sb..contents & sa..size = sb..size"
95          ensures "True" */
96      {
97          /*: assume "True" */
98          boolean r1a = sa.containsKey(k1);
99          Object r2a = sa.get(k2);
100
101         Object r2b = sb.get(k2);
102         boolean r1b = sb.containsKey(k1);
103
104         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
105     }
106
107     static void containsKey_get_pre_c_3(AssociationList sa, AssociationList sb, Object
            k1, Object k2)
108     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
109                 sa..contents = sb..contents & sa..size = sb..size"
110         ensures "True" */
111     {
112         /*: assume "~(True)" */
113         boolean r1a = sa.containsKey(k1);
114         Object r2a = sa.get(k2);
115
116         Object r2b = sb.get(k2);
117         boolean r1b = sb.containsKey(k1);
118
119         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
120     }
121
122     static void containsKey_get_between_s_4(AssociationList sa, AssociationList sb,
            Object k1, Object k2)
123     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
124                 sa..contents = sb..contents & sa..size = sb..size"
125         ensures "True" */
```

```
126     {
127         boolean r1a = sa.containsKey(k1);
128         /*: assume "True" */
129         Object r2a = sa.get(k2);
130 
131         Object r2b = sb.get(k2);
132         boolean r1b = sb.containsKey(k1);
133 
134         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
135     }
136 
137     static void containsKey_get_between_c_4(AssociationList sa, AssociationList sb,
            Object k1, Object k2)
138     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
139                   sa..contents = sb..contents & sa..size = sb..size"
140         ensures "True" */
141     {
142         boolean r1a = sa.containsKey(k1);
143         /*: assume "~(True)" */
144         Object r2a = sa.get(k2);
145 
146         Object r2b = sb.get(k2);
147         boolean r1b = sb.containsKey(k1);
148 
149         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
150     }
151 
152     static void containsKey_get_post_s_5(AssociationList sa, AssociationList sb, Object
            k1, Object k2)
153     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
154                   sa..contents = sb..contents & sa..size = sb..size"
155         ensures "True" */
156     {
157         boolean r1a = sa.containsKey(k1);
158         Object r2a = sa.get(k2);
159         /*: assume "True" */
160 
161         Object r2b = sb.get(k2);
162         boolean r1b = sb.containsKey(k1);
163 
164         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
165     }
166 
167     static void containsKey_get_post_c_5(AssociationList sa, AssociationList sb, Object
            k1, Object k2)
168     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
169                   sa..contents = sb..contents & sa..size = sb..size"
170         ensures "True" */
171     {
172         boolean r1a = sa.containsKey(k1);
173         Object r2a = sa.get(k2);
174         /*: assume "~(True)" */
175 
176         Object r2b = sb.get(k2);
177         boolean r1b = sb.containsKey(k1);
178 
179         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
180     }
181 
182     static void containsKey_put_pre_s_6(AssociationList sa, AssociationList sb, Object
            k1, Object k2, Object v2)
```

```
183     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
184                 sa..contents = sb..contents & sa..size = sb..size"
185         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
186         ensures "True" */
187     {
188         /*: assume "k1 ~= k2 | (EX v. (k1, v) : sa..contents)" */
189         boolean r1a = sa.containsKey(k1);
190         Object r2a = sa.put(k2, v2);
191 
192         Object r2b = sb.put(k2, v2);
193         boolean r1b = sb.containsKey(k1);
194 
195         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
196     }
197 
198     static void containsKey_put_pre_c_6(AssociationList sa, AssociationList sb, Object
            k1, Object k2, Object v2)
199     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
200                 sa..contents = sb..contents & sa..size = sb..size"
201         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
202         ensures "True" */
203     {
204         /*: assume "~(k1 ~= k2 | (EX v. (k1, v) : sa..contents))" */
205         boolean r1a = sa.containsKey(k1);
206         Object r2a = sa.put(k2, v2);
207 
208         Object r2b = sb.put(k2, v2);
209         boolean r1b = sb.containsKey(k1);
210 
211         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
212     }
213 
214     static void containsKey_put_between_s_7(AssociationList sa, AssociationList sb,
            Object k1, Object k2, Object v2)
215     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
216                 sa..contents = sb..contents & sa..size = sb..size"
217         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
218         ensures "True" */
219     {
220         boolean r1a = sa.containsKey(k1);
221         /*: assume "k1 ~= k2 | r1a" */
222         Object r2a = sa.put(k2, v2);
223 
224         Object r2b = sb.put(k2, v2);
225         boolean r1b = sb.containsKey(k1);
226 
227         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
228     }
229 
230     static void containsKey_put_between_c_7(AssociationList sa, AssociationList sb,
            Object k1, Object k2, Object v2)
231     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
232                 sa..contents = sb..contents & sa..size = sb..size"
233         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
234         ensures "True" */
235     {
236         boolean r1a = sa.containsKey(k1);
237         /*: assume "~(k1 ~= k2 | r1a)" */
238         Object r2a = sa.put(k2, v2);
239 
240         Object r2b = sb.put(k2, v2);
241         boolean r1b = sb.containsKey(k1);
```

```
242
243         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
244     }
245
246     static void containsKey_put_post_s_8(AssociationList sa, AssociationList sb, Object
            k1, Object k2, Object v2)
247     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
248                   sa..contents = sb..contents & sa..size = sb..size"
249         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
250         ensures "True" */
251     {
252         boolean r1a = sa.containsKey(k1);
253         Object r2a = sa.put(k2, v2);
254         /*: assume "k1 ~= k2 | r1a" */
255
256         Object r2b = sb.put(k2, v2);
257         boolean r1b = sb.containsKey(k1);
258
259         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
260     }
261
262     static void containsKey_put_post_c_8(AssociationList sa, AssociationList sb, Object
            k1, Object k2, Object v2)
263     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
264                   sa..contents = sb..contents & sa..size = sb..size"
265         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
266         ensures "True" */
267     {
268         boolean r1a = sa.containsKey(k1);
269         Object r2a = sa.put(k2, v2);
270         /*: assume "~(k1 ~= k2 | r1a)" */
271
272         Object r2b = sb.put(k2, v2);
273         boolean r1b = sb.containsKey(k1);
274
275         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
276     }
277
278     static void containsKey_put_pre_s_9(AssociationList sa, AssociationList sb, Object
            k1, Object k2, Object v2)
279     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
280                   sa..contents = sb..contents & sa..size = sb..size"
281         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
282         ensures "True" */
283     {
284         /*: assume "k1 ~= k2 | (EX v. (k1, v) : sa..contents)" */
285         boolean r1a = sa.containsKey(k1);
286         sa.put(k2, v2);
287
288         sb.put(k2, v2);
289         boolean r1b = sb.containsKey(k1);
290
291         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
292     }
293
294     static void containsKey_put_pre_c_9(AssociationList sa, AssociationList sb, Object
            k1, Object k2, Object v2)
295     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
296                   sa..contents = sb..contents & sa..size = sb..size"
297         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
298         ensures "True" */
299     {
```

```
        /*: assume "~(k1 ~= k2 | (EX v. (k1, v) : sa..contents))" */
        boolean r1a = sa.containsKey(k1);
        sa.put(k2, v2);

        sb.put(k2, v2);
        boolean r1b = sb.containsKey(k1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void containsKey_put_between_s_10(AssociationList sa, AssociationList sb,
        Object k1, Object k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.containsKey(k1);
        /*: assume "k1 ~= k2 | r1a" */
        sa.put(k2, v2);

        sb.put(k2, v2);
        boolean r1b = sb.containsKey(k1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void containsKey_put_between_c_10(AssociationList sa, AssociationList sb,
        Object k1, Object k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.containsKey(k1);
        /*: assume "~(k1 ~= k2 | r1a)" */
        sa.put(k2, v2);

        sb.put(k2, v2);
        boolean r1b = sb.containsKey(k1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void containsKey_put_post_s_11(AssociationList sa, AssociationList sb,
        Object k1, Object k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.containsKey(k1);
        sa.put(k2, v2);
        /*: assume "k1 ~= k2 | r1a" */

        sb.put(k2, v2);
        boolean r1b = sb.containsKey(k1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void containsKey_put_post_c_11(AssociationList sa, AssociationList sb,
        Object k1, Object k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
```

```
361         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
362         ensures "True" */
363     {
364         boolean r1a = sa.containsKey(k1);
365         sa.put(k2, v2);
366         /*: assume "~(k1 ~= k2 | r1a)" */
367 
368         sb.put(k2, v2);
369         boolean r1b = sb.containsKey(k1);
370 
371         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
372     }
373 
374     static void containsKey_remove_pre_s_12(AssociationList sa, AssociationList sb,
        Object k1, Object k2)
375     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
376                 sa..contents = sb..contents & sa..size = sb..size"
377         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
378         ensures "True" */
379     {
380         /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
381         boolean r1a = sa.containsKey(k1);
382         Object r2a = sa.remove(k2);
383 
384         Object r2b = sb.remove(k2);
385         boolean r1b = sb.containsKey(k1);
386 
387         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
388     }
389 
390     static void containsKey_remove_pre_c_12(AssociationList sa, AssociationList sb,
        Object k1, Object k2)
391     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
392                 sa..contents = sb..contents & sa..size = sb..size"
393         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
394         ensures "True" */
395     {
396         /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
397         boolean r1a = sa.containsKey(k1);
398         Object r2a = sa.remove(k2);
399 
400         Object r2b = sb.remove(k2);
401         boolean r1b = sb.containsKey(k1);
402 
403         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
404     }
405 
406     static void containsKey_remove_between_s_13(AssociationList sa, AssociationList sb,
        Object k1, Object k2)
407     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
408                 sa..contents = sb..contents & sa..size = sb..size"
409         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
410         ensures "True" */
411     {
412         boolean r1a = sa.containsKey(k1);
413         /*: assume "k1 ~= k2 | ~r1a" */
414         Object r2a = sa.remove(k2);
415 
416         Object r2b = sb.remove(k2);
417         boolean r1b = sb.containsKey(k1);
418 
419         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
```

```
420     }
421
422     static void containsKey_remove_between_c_13(AssociationList sa, AssociationList sb,
            Object k1, Object k2)
423     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
424                    sa..contents = sb..contents & sa..size = sb..size"
425         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
426         ensures "True" */
427     {
428         boolean r1a = sa.containsKey(k1);
429         /*: assume "~(k1 ~= k2 | ~r1a)" */
430         Object r2a = sa.remove(k2);
431
432         Object r2b = sb.remove(k2);
433         boolean r1b = sb.containsKey(k1);
434
435         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
436     }
437
438     static void containsKey_remove_post_s_14(AssociationList sa, AssociationList sb,
            Object k1, Object k2)
439     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
440                    sa..contents = sb..contents & sa..size = sb..size"
441         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
442         ensures "True" */
443     {
444         boolean r1a = sa.containsKey(k1);
445         Object r2a = sa.remove(k2);
446         /*: assume "k1 ~= k2 | ~r1a" */
447
448         Object r2b = sb.remove(k2);
449         boolean r1b = sb.containsKey(k1);
450
451         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
452     }
453
454     static void containsKey_remove_post_c_14(AssociationList sa, AssociationList sb,
            Object k1, Object k2)
455     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
456                    sa..contents = sb..contents & sa..size = sb..size"
457         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
458         ensures "True" */
459     {
460         boolean r1a = sa.containsKey(k1);
461         Object r2a = sa.remove(k2);
462         /*: assume "~(k1 ~= k2 | ~r1a)" */
463
464         Object r2b = sb.remove(k2);
465         boolean r1b = sb.containsKey(k1);
466
467         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
468     }
469
470     static void containsKey_remove_pre_s_15(AssociationList sa, AssociationList sb,
            Object k1, Object k2)
471     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
472                    sa..contents = sb..contents & sa..size = sb..size"
473         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
474         ensures "True" */
475     {
476         /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
477         boolean r1a = sa.containsKey(k1);
```

```
478         sa.remove(k2);
479
480         sb.remove(k2);
481         boolean r1b = sb.containsKey(k1);
482
483         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
484     }
485
486     static void containsKey_remove_pre_c_15(AssociationList sa, AssociationList sb,
        Object k1, Object k2)
487     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
488                   sa..contents = sb..contents & sa..size = sb..size"
489         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
490         ensures "True" */
491     {
492         /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
493         boolean r1a = sa.containsKey(k1);
494         sa.remove(k2);
495
496         sb.remove(k2);
497         boolean r1b = sb.containsKey(k1);
498
499         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
500     }
501
502     static void containsKey_remove_between_s_16(AssociationList sa, AssociationList sb,
        Object k1, Object k2)
503     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
504                   sa..contents = sb..contents & sa..size = sb..size"
505         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
506         ensures "True" */
507     {
508         boolean r1a = sa.containsKey(k1);
509         /*: assume "k1 ~= k2 | ~r1a" */
510         sa.remove(k2);
511
512         sb.remove(k2);
513         boolean r1b = sb.containsKey(k1);
514
515         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
516     }
517
518     static void containsKey_remove_between_c_16(AssociationList sa, AssociationList sb,
        Object k1, Object k2)
519     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
520                   sa..contents = sb..contents & sa..size = sb..size"
521         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
522         ensures "True" */
523     {
524         boolean r1a = sa.containsKey(k1);
525         /*: assume "~(k1 ~= k2 | ~r1a)" */
526         sa.remove(k2);
527
528         sb.remove(k2);
529         boolean r1b = sb.containsKey(k1);
530
531         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
532     }
533
534     static void containsKey_remove_post_s_17(AssociationList sa, AssociationList sb,
        Object k1, Object k2)
535     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
536                   sa..contents = sb..contents & sa..size = sb..size"
537         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
538         ensures "True" */
```

```
539     {
540         boolean r1a = sa.containsKey(k1);
541         sa.remove(k2);
542         /*: assume "k1 ~= k2 | ~r1a" */
543 
544         sb.remove(k2);
545         boolean r1b = sb.containsKey(k1);
546 
547         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
548     }
549 
550     static void containsKey_remove_post_c_17(AssociationList sa, AssociationList sb,
        Object k1, Object k2)
551     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
552                   sa..contents = sb..contents & sa..size = sb..size"
553         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
554         ensures "True" */
555     {
556         boolean r1a = sa.containsKey(k1);
557         sa.remove(k2);
558         /*: assume "~(k1 ~= k2 | ~r1a)" */
559 
560         sb.remove(k2);
561         boolean r1b = sb.containsKey(k1);
562 
563         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
564     }
565 
566     static void containsKey_size_pre_s_18(AssociationList sa, AssociationList sb,
        Object k1)
567     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
568                   sa..contents = sb..contents & sa..size = sb..size"
569         ensures "True" */
570     {
571         /*: assume "True" */
572         boolean r1a = sa.containsKey(k1);
573         int r2a = sa.size();
574 
575         int r2b = sb.size();
576         boolean r1b = sb.containsKey(k1);
577 
578         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
579     }
580 
581     static void containsKey_size_pre_c_18(AssociationList sa, AssociationList sb,
        Object k1)
582     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
583                   sa..contents = sb..contents & sa..size = sb..size"
584         ensures "True" */
585     {
586         /*: assume "~(True)" */
587         boolean r1a = sa.containsKey(k1);
588         int r2a = sa.size();
589 
590         int r2b = sb.size();
591         boolean r1b = sb.containsKey(k1);
592 
593         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
594     }
595 
596     static void containsKey_size_between_s_19(AssociationList sa, AssociationList sb,
        Object k1)
597     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
```

```
598                 sa..contents = sb..contents & sa..size = sb..size"
599         ensures "True" */
600     {
601         boolean r1a = sa.containsKey(k1);
602         /*: assume "True" */
603         int r2a = sa.size();
604 
605         int r2b = sb.size();
606         boolean r1b = sb.containsKey(k1);
607 
608         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
609     }
610 
611     static void containsKey_size_between_c_19(AssociationList sa, AssociationList sb,
            Object k1)
612     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
613                 sa..contents = sb..contents & sa..size = sb..size"
614         ensures "True" */
615     {
616         boolean r1a = sa.containsKey(k1);
617         /*: assume "~(True)" */
618         int r2a = sa.size();
619 
620         int r2b = sb.size();
621         boolean r1b = sb.containsKey(k1);
622 
623         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
624     }
625 
626     static void containsKey_size_post_s_20(AssociationList sa, AssociationList sb,
            Object k1)
627     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
628                 sa..contents = sb..contents & sa..size = sb..size"
629         ensures "True" */
630     {
631         boolean r1a = sa.containsKey(k1);
632         int r2a = sa.size();
633         /*: assume "True" */
634 
635         int r2b = sb.size();
636         boolean r1b = sb.containsKey(k1);
637 
638         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
639     }
640 
641     static void containsKey_size_post_c_20(AssociationList sa, AssociationList sb,
            Object k1)
642     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
643                 sa..contents = sb..contents & sa..size = sb..size"
644         ensures "True" */
645     {
646         boolean r1a = sa.containsKey(k1);
647         int r2a = sa.size();
648         /*: assume "~(True)" */
649 
650         int r2b = sb.size();
651         boolean r1b = sb.containsKey(k1);
652 
653         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
654     }
655 
```

```
656     static void get_containsKey_pre_s_21(AssociationList sa, AssociationList sb, Object
            k1, Object k2)
657     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
658                 sa..contents = sb..contents & sa..size = sb..size"
659         ensures "True" */
660     {
661         /*: assume "True" */
662         Object r1a = sa.get(k1);
663         boolean r2a = sa.containsKey(k2);
664 
665         boolean r2b = sb.containsKey(k2);
666         Object r1b = sb.get(k1);
667 
668         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
669     }
670 
671     static void get_containsKey_pre_c_21(AssociationList sa, AssociationList sb, Object
            k1, Object k2)
672     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
673                 sa..contents = sb..contents & sa..size = sb..size"
674         ensures "True" */
675     {
676         /*: assume "~(True)" */
677         Object r1a = sa.get(k1);
678         boolean r2a = sa.containsKey(k2);
679 
680         boolean r2b = sb.containsKey(k2);
681         Object r1b = sb.get(k1);
682 
683         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
684     }
685 
686     static void get_containsKey_between_s_22(AssociationList sa, AssociationList sb,
            Object k1, Object k2)
687     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
688                 sa..contents = sb..contents & sa..size = sb..size"
689         ensures "True" */
690     {
691         Object r1a = sa.get(k1);
692         /*: assume "True" */
693         boolean r2a = sa.containsKey(k2);
694 
695         boolean r2b = sb.containsKey(k2);
696         Object r1b = sb.get(k1);
697 
698         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
699     }
700 
701     static void get_containsKey_between_c_22(AssociationList sa, AssociationList sb,
            Object k1, Object k2)
702     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
703                 sa..contents = sb..contents & sa..size = sb..size"
704         ensures "True" */
705     {
706         Object r1a = sa.get(k1);
707         /*: assume "~(True)" */
708         boolean r2a = sa.containsKey(k2);
709 
710         boolean r2b = sb.containsKey(k2);
711         Object r1b = sb.get(k1);
712 
```

```
713         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
714     }
715
716     static void get_containsKey_post_s_23(AssociationList sa, AssociationList sb,
            Object k1, Object k2)
717     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
718                   sa..contents = sb..contents & sa..size = sb..size"
719         ensures "True" */
720     {
721         Object r1a = sa.get(k1);
722         boolean r2a = sa.containsKey(k2);
723         /*: assume "True" */
724
725         boolean r2b = sb.containsKey(k2);
726         Object r1b = sb.get(k1);
727
728         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
729     }
730
731     static void get_containsKey_post_c_23(AssociationList sa, AssociationList sb,
            Object k1, Object k2)
732     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
733                   sa..contents = sb..contents & sa..size = sb..size"
734         ensures "True" */
735     {
736         Object r1a = sa.get(k1);
737         boolean r2a = sa.containsKey(k2);
738         /*: assume "~(True)" */
739
740         boolean r2b = sb.containsKey(k2);
741         Object r1b = sb.get(k1);
742
743         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
744     }
745
746     static void get_get_pre_s_24(AssociationList sa, AssociationList sb, Object k1,
            Object k2)
747     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
748                   sa..contents = sb..contents & sa..size = sb..size"
749         ensures "True" */
750     {
751         /*: assume "True" */
752         Object r1a = sa.get(k1);
753         Object r2a = sa.get(k2);
754
755         Object r2b = sb.get(k2);
756         Object r1b = sb.get(k1);
757
758         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
759     }
760
761     static void get_get_pre_c_24(AssociationList sa, AssociationList sb, Object k1,
            Object k2)
762     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
763                   sa..contents = sb..contents & sa..size = sb..size"
764         ensures "True" */
765     {
766         /*: assume "~(True)" */
767         Object r1a = sa.get(k1);
768         Object r2a = sa.get(k2);
769
```

```
        Object r2b = sb.get(k2);
        Object r1b = sb.get(k1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void get_get_between_s_25(AssociationList sa, AssociationList sb, Object k1,
        Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        Object r1a = sa.get(k1);
        /*: assume "True" */
        Object r2a = sa.get(k2);

        Object r2b = sb.get(k2);
        Object r1b = sb.get(k1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void get_get_between_c_25(AssociationList sa, AssociationList sb, Object k1,
        Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        Object r1a = sa.get(k1);
        /*: assume "~(True)" */
        Object r2a = sa.get(k2);

        Object r2b = sb.get(k2);
        Object r1b = sb.get(k1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void get_get_post_s_26(AssociationList sa, AssociationList sb, Object k1,
        Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        Object r1a = sa.get(k1);
        Object r2a = sa.get(k2);
        /*: assume "True" */

        Object r2b = sb.get(k2);
        Object r1b = sb.get(k1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void get_get_post_c_26(AssociationList sa, AssociationList sb, Object k1,
        Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        Object r1a = sa.get(k1);
```

```
827         Object r2a = sa.get(k2);
828         /*: assume "~(True)" */
829 
830         Object r2b = sb.get(k2);
831         Object r1b = sb.get(k1);
832 
833         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
834     }
835 
836     static void get_put_pre_s_27(AssociationList sa, AssociationList sb, Object k1,
            Object k2, Object v2)
837     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
            null &
838                 sa..contents = sb..contents & sa..size = sb..size"
839         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
840         ensures "True" */
841     {
842         /*: assume "k1 ~= k2 | (k1, v2) : sa..contents" */
843         Object r1a = sa.get(k1);
844         Object r2a = sa.put(k2, v2);
845 
846         Object r2b = sb.put(k2, v2);
847         Object r1b = sb.get(k1);
848 
849         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
850     }
851 
852     static void get_put_pre_c_27(AssociationList sa, AssociationList sb, Object k1,
            Object k2, Object v2)
853     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
            null &
854                 sa..contents = sb..contents & sa..size = sb..size"
855         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
856         ensures "True" */
857     {
858         /*: assume "~(k1 ~= k2 | (k1, v2) : sa..contents)" */
859         Object r1a = sa.get(k1);
860         Object r2a = sa.put(k2, v2);
861 
862         Object r2b = sb.put(k2, v2);
863         Object r1b = sb.get(k1);
864 
865         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
866     }
867 
868     static void get_put_between_s_28(AssociationList sa, AssociationList sb, Object k1,
            Object k2, Object v2)
869     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
            null &
870                 sa..contents = sb..contents & sa..size = sb..size"
871         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
872         ensures "True" */
873     {
874         Object r1a = sa.get(k1);
875         /*: assume "k1 ~= k2 | r1a = v2" */
876         Object r2a = sa.put(k2, v2);
877 
878         Object r2b = sb.put(k2, v2);
879         Object r1b = sb.get(k1);
880 
881         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
```

```
882     }
883
884     static void get_put_between_c_28(AssociationList sa, AssociationList sb, Object k1,
            Object k2, Object v2)
885     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
            null &
886                 sa..contents = sb..contents & sa..size = sb..size"
887         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
888         ensures "True" */
889     {
890         Object r1a = sa.get(k1);
891         /*: assume "~(k1 ~= k2 | r1a = v2)" */
892         Object r2a = sa.put(k2, v2);
893
894         Object r2b = sb.put(k2, v2);
895         Object r1b = sb.get(k1);
896
897         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
898     }
899
900     static void get_put_post_s_29(AssociationList sa, AssociationList sb, Object k1,
            Object k2, Object v2)
901     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
            null &
902                 sa..contents = sb..contents & sa..size = sb..size"
903         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
904         ensures "True" */
905     {
906         Object r1a = sa.get(k1);
907         Object r2a = sa.put(k2, v2);
908         /*: assume "k1 ~= k2 | r1a = v2" */
909
910         Object r2b = sb.put(k2, v2);
911         Object r1b = sb.get(k1);
912
913         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
914     }
915
916     static void get_put_post_c_29(AssociationList sa, AssociationList sb, Object k1,
            Object k2, Object v2)
917     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
            null &
918                 sa..contents = sb..contents & sa..size = sb..size"
919         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
920         ensures "True" */
921     {
922         Object r1a = sa.get(k1);
923         Object r2a = sa.put(k2, v2);
924         /*: assume "~(k1 ~= k2 | r1a = v2)" */
925
926         Object r2b = sb.put(k2, v2);
927         Object r1b = sb.get(k1);
928
929         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
930     }
931
932     static void get_put_pre_s_30(AssociationList sa, AssociationList sb, Object k1,
            Object k2, Object v2)
933     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
            null &
934                 sa..contents = sb..contents & sa..size = sb..size"
935         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
```

```
936         ensures "True" */
937     {
938         /*: assume "k1 ~= k2 | (k1, v2) : sa..contents" */
939         Object r1a = sa.get(k1);
940         sa.put(k2, v2);
941 
942         sb.put(k2, v2);
943         Object r1b = sb.get(k1);
944 
945         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
946     }
947 
948     static void get_put_pre_c_30(AssociationList sa, AssociationList sb, Object k1,
        Object k2, Object v2)
949     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
        null &
950                 sa..contents = sb..contents & sa..size = sb..size"
951         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
952         ensures "True" */
953     {
954         /*: assume "~(k1 ~= k2 | (k1, v2) : sa..contents)" */
955         Object r1a = sa.get(k1);
956         sa.put(k2, v2);
957 
958         sb.put(k2, v2);
959         Object r1b = sb.get(k1);
960 
961         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
962     }
963 
964     static void get_put_between_s_31(AssociationList sa, AssociationList sb, Object k1,
        Object k2, Object v2)
965     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
        null &
966                 sa..contents = sb..contents & sa..size = sb..size"
967         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
968         ensures "True" */
969     {
970         Object r1a = sa.get(k1);
971         /*: assume "k1 ~= k2 | r1a = v2" */
972         sa.put(k2, v2);
973 
974         sb.put(k2, v2);
975         Object r1b = sb.get(k1);
976 
977         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
978     }
979 
980     static void get_put_between_c_31(AssociationList sa, AssociationList sb, Object k1,
        Object k2, Object v2)
981     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
        null &
982                 sa..contents = sb..contents & sa..size = sb..size"
983         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
984         ensures "True" */
985     {
986         Object r1a = sa.get(k1);
987         /*: assume "~(k1 ~= k2 | r1a = v2)" */
988         sa.put(k2, v2);
989 
990         sb.put(k2, v2);
991         Object r1b = sb.get(k1);
992 
993         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
994     }
```

```
995
996     static void get_put_post_s_32(AssociationList sa, AssociationList sb, Object k1,
            Object k2, Object v2)
997     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
            null &
998                 sa..contents = sb..contents & sa..size = sb..size"
999         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1000        ensures "True" */
1001    {
1002        Object r1a = sa.get(k1);
1003        sa.put(k2, v2);
1004        /*: assume "k1 ~= k2 | r1a = v2" */
1005
1006        sb.put(k2, v2);
1007        Object r1b = sb.get(k1);
1008
1009        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1010    }
1011
1012    static void get_put_post_c_32(AssociationList sa, AssociationList sb, Object k1,
            Object k2, Object v2)
1013    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
            null &
1014                sa..contents = sb..contents & sa..size = sb..size"
1015        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1016        ensures "True" */
1017    {
1018        Object r1a = sa.get(k1);
1019        sa.put(k2, v2);
1020        /*: assume "~(k1 ~= k2 | r1a = v2)" */
1021
1022        sb.put(k2, v2);
1023        Object r1b = sb.get(k1);
1024
1025        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1026    }
1027
1028    static void get_remove_pre_s_33(AssociationList sa, AssociationList sb, Object k1,
            Object k2)
1029    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
1030                sa..contents = sb..contents & sa..size = sb..size"
1031        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1032        ensures "True" */
1033    {
1034        /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
1035        Object r1a = sa.get(k1);
1036        Object r2a = sa.remove(k2);
1037
1038        Object r2b = sb.remove(k2);
1039        Object r1b = sb.get(k1);
1040
1041        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
1042    }
1043
1044    static void get_remove_pre_c_33(AssociationList sa, AssociationList sb, Object k1,
            Object k2)
1045    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
1046                sa..contents = sb..contents & sa..size = sb..size"
1047        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1048        ensures "True" */
1049    {
1050        /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
1051        Object r1a = sa.get(k1);
1052        Object r2a = sa.remove(k2);
```

```
1053
1054            Object r2b = sb.remove(k2);
1055            Object r1b = sb.get(k1);
1056
1057            /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                    sb..size)" */
1058        }
1059
1060        static void get_remove_between_s_34(AssociationList sa, AssociationList sb, Object
                k1, Object k2)
1061        /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
1062                      sa..contents = sb..contents & sa..size = sb..size"
1063            modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1064            ensures "True" */
1065        {
1066            Object r1a = sa.get(k1);
1067            /*: assume "k1 ~= k2 | r1a = null" */
1068            Object r2a = sa.remove(k2);
1069
1070            Object r2b = sb.remove(k2);
1071            Object r1b = sb.get(k1);
1072
1073            /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                    sb..size" */
1074        }
1075
1076        static void get_remove_between_c_34(AssociationList sa, AssociationList sb, Object
                k1, Object k2)
1077        /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
1078                      sa..contents = sb..contents & sa..size = sb..size"
1079            modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1080            ensures "True" */
1081        {
1082            Object r1a = sa.get(k1);
1083            /*: assume "~(k1 ~= k2 | r1a = null)" */
1084            Object r2a = sa.remove(k2);
1085
1086            Object r2b = sb.remove(k2);
1087            Object r1b = sb.get(k1);
1088
1089            /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                    sb..size)" */
1090        }
1091
1092        static void get_remove_post_s_35(AssociationList sa, AssociationList sb, Object k1,
                Object k2)
1093        /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
1094                      sa..contents = sb..contents & sa..size = sb..size"
1095            modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1096            ensures "True" */
1097        {
1098            Object r1a = sa.get(k1);
1099            Object r2a = sa.remove(k2);
1100            /*: assume "k1 ~= k2 | r1a = null" */
1101
1102            Object r2b = sb.remove(k2);
1103            Object r1b = sb.get(k1);
1104
1105            /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                    sb..size" */
1106        }
1107
1108        static void get_remove_post_c_35(AssociationList sa, AssociationList sb, Object k1,
                Object k2)
1109        /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
```

```
1110                 sa..contents = sb..contents & sa..size = sb..size"
1111         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1112         ensures "True" */
1113     {
1114         Object r1a = sa.get(k1);
1115         Object r2a = sa.remove(k2);
1116         /*: assume "~(k1 ~= k2 | r1a = null)" */
1117 
1118         Object r2b = sb.remove(k2);
1119         Object r1b = sb.get(k1);
1120 
1121         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
1122     }
1123 
1124     static void get_remove_pre_s_36(AssociationList sa, AssociationList sb, Object k1,
            Object k2)
1125     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
1126                 sa..contents = sb..contents & sa..size = sb..size"
1127         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1128         ensures "True" */
1129     {
1130         /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
1131         Object r1a = sa.get(k1);
1132         sa.remove(k2);
1133 
1134         sb.remove(k2);
1135         Object r1b = sb.get(k1);
1136 
1137         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1138     }
1139 
1140     static void get_remove_pre_c_36(AssociationList sa, AssociationList sb, Object k1,
            Object k2)
1141     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
1142                 sa..contents = sb..contents & sa..size = sb..size"
1143         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1144         ensures "True" */
1145     {
1146         /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
1147         Object r1a = sa.get(k1);
1148         sa.remove(k2);
1149 
1150         sb.remove(k2);
1151         Object r1b = sb.get(k1);
1152 
1153         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1154     }
1155 
1156     static void get_remove_between_s_37(AssociationList sa, AssociationList sb, Object
            k1, Object k2)
1157     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
1158                 sa..contents = sb..contents & sa..size = sb..size"
1159         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1160         ensures "True" */
1161     {
1162         Object r1a = sa.get(k1);
1163         /*: assume "k1 ~= k2 | r1a = null" */
1164         sa.remove(k2);
1165 
1166         sb.remove(k2);
1167         Object r1b = sb.get(k1);
1168 
1169         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1170     }
```

```
1171
1172    static void get_remove_between_c_37(AssociationList sa, AssociationList sb, Object
            k1, Object k2)
1173    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
1174                 sa..contents = sb..contents & sa..size = sb..size"
1175        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1176        ensures "True" */
1177    {
1178        Object r1a = sa.get(k1);
1179        /*: assume "~(k1 ~= k2 | r1a = null)" */
1180        sa.remove(k2);
1181
1182        sb.remove(k2);
1183        Object r1b = sb.get(k1);
1184
1185        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1186    }
1187
1188    static void get_remove_post_s_38(AssociationList sa, AssociationList sb, Object k1,
            Object k2)
1189    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
1190                 sa..contents = sb..contents & sa..size = sb..size"
1191        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1192        ensures "True" */
1193    {
1194        Object r1a = sa.get(k1);
1195        sa.remove(k2);
1196        /*: assume "k1 ~= k2 | r1a = null" */
1197
1198        sb.remove(k2);
1199        Object r1b = sb.get(k1);
1200
1201        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1202    }
1203
1204    static void get_remove_post_c_38(AssociationList sa, AssociationList sb, Object k1,
            Object k2)
1205    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
1206                 sa..contents = sb..contents & sa..size = sb..size"
1207        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1208        ensures "True" */
1209    {
1210        Object r1a = sa.get(k1);
1211        sa.remove(k2);
1212        /*: assume "~(k1 ~= k2 | r1a = null)" */
1213
1214        sb.remove(k2);
1215        Object r1b = sb.get(k1);
1216
1217        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1218    }
1219
1220    static void get_size_pre_s_39(AssociationList sa, AssociationList sb, Object k1)
1221    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
1222                 sa..contents = sb..contents & sa..size = sb..size"
1223        ensures "True" */
1224    {
1225        /*: assume "True" */
1226        Object r1a = sa.get(k1);
1227        int r2a = sa.size();
1228
1229        int r2b = sb.size();
1230        Object r1b = sb.get(k1);
1231
```

```
        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void get_size_pre_c_39(AssociationList sa, AssociationList sb, Object k1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        /*: assume "~(True)" */
        Object r1a = sa.get(k1);
        int r2a = sa.size();

        int r2b = sb.size();
        Object r1b = sb.get(k1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void get_size_between_s_40(AssociationList sa, AssociationList sb, Object k1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        Object r1a = sa.get(k1);
        /*: assume "True" */
        int r2a = sa.size();

        int r2b = sb.size();
        Object r1b = sb.get(k1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void get_size_between_c_40(AssociationList sa, AssociationList sb, Object k1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        Object r1a = sa.get(k1);
        /*: assume "~(True)" */
        int r2a = sa.size();

        int r2b = sb.size();
        Object r1b = sb.get(k1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void get_size_post_s_41(AssociationList sa, AssociationList sb, Object k1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        Object r1a = sa.get(k1);
        int r2a = sa.size();
        /*: assume "True" */

        int r2b = sb.size();
        Object r1b = sb.get(k1);

```

```
        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void get_size_post_c_41(AssociationList sa, AssociationList sb, Object k1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        Object r1a = sa.get(k1);
        int r2a = sa.size();
        /*: assume "~(True)" */

        int r2b = sb.size();
        Object r1b = sb.get(k1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void put_containsKey_pre_s_42(AssociationList sa, AssociationList sb, Object
        k1, Object v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "k1 ~= k2 | (EX v. (k1, v) : sa..contents)" */
        Object r1a = sa.put(k1, v1);
        boolean r2a = sa.containsKey(k2);

        boolean r2b = sb.containsKey(k2);
        Object r1b = sb.put(k1, v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void put_containsKey_pre_c_42(AssociationList sa, AssociationList sb, Object
        k1, Object v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(k1 ~= k2 | (EX v. (k1, v) : sa..contents))" */
        Object r1a = sa.put(k1, v1);
        boolean r2a = sa.containsKey(k2);

        boolean r2b = sb.containsKey(k2);
        Object r1b = sb.put(k1, v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void put_containsKey_between_s_43(AssociationList sa, AssociationList sb,
        Object k1, Object v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.put(k1, v1);
        /*: assume "k1 ~= k2 | r1a ~= null" */
```

```
1350        boolean r2a = sa.containsKey(k2);
1351
1352        boolean r2b = sb.containsKey(k2);
1353        Object r1b = sb.put(k1, v1);
1354
1355        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
1356    }
1357
1358    static void put_containsKey_between_c_43(AssociationList sa, AssociationList sb,
            Object k1, Object v1, Object k2)
1359    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
1360                  sa..contents = sb..contents & sa..size = sb..size"
1361        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1362        ensures "True" */
1363    {
1364        Object r1a = sa.put(k1, v1);
1365        /*: assume "~(k1 ~= k2 | r1a ~= null)" */
1366        boolean r2a = sa.containsKey(k2);
1367
1368        boolean r2b = sb.containsKey(k2);
1369        Object r1b = sb.put(k1, v1);
1370
1371        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
1372    }
1373
1374    static void put_containsKey_post_s_44(AssociationList sa, AssociationList sb,
            Object k1, Object v1, Object k2)
1375    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
1376                  sa..contents = sb..contents & sa..size = sb..size"
1377        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1378        ensures "True" */
1379    {
1380        Object r1a = sa.put(k1, v1);
1381        boolean r2a = sa.containsKey(k2);
1382        /*: assume "k1 ~= k2 | r1a ~= null" */
1383
1384        boolean r2b = sb.containsKey(k2);
1385        Object r1b = sb.put(k1, v1);
1386
1387        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
1388    }
1389
1390    static void put_containsKey_post_c_44(AssociationList sa, AssociationList sb,
            Object k1, Object v1, Object k2)
1391    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
1392                  sa..contents = sb..contents & sa..size = sb..size"
1393        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1394        ensures "True" */
1395    {
1396        Object r1a = sa.put(k1, v1);
1397        boolean r2a = sa.containsKey(k2);
1398        /*: assume "~(k1 ~= k2 | r1a ~= null)" */
1399
1400        boolean r2b = sb.containsKey(k2);
1401        Object r1b = sb.put(k1, v1);
1402
1403        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
1404    }
1405
1406    static void put_get_pre_s_45(AssociationList sa, AssociationList sb, Object k1,
            Object v1, Object k2)
```

```
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "k1 ~= k2 | (k1, v1) : sa..contents" */
        Object r1a = sa.put(k1, v1);
        Object r2a = sa.get(k2);

        Object r2b = sb.get(k2);
        Object r1b = sb.put(k1, v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void put_get_pre_c_45(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(k1 ~= k2 | (k1, v1) : sa..contents)" */
        Object r1a = sa.put(k1, v1);
        Object r2a = sa.get(k2);

        Object r2b = sb.get(k2);
        Object r1b = sb.put(k1, v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void put_get_between_s_46(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.put(k1, v1);
        /*: assume "k1 ~= k2 | r1a = v1" */
        Object r2a = sa.get(k2);

        Object r2b = sb.get(k2);
        Object r1b = sb.put(k1, v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void put_get_between_c_46(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.put(k1, v1);
        /*: assume "~(k1 ~= k2 | r1a = v1)" */
```

```
1462        Object r2a = sa.get(k2);
1463
1464        Object r2b = sb.get(k2);
1465        Object r1b = sb.put(k1, v1);
1466
1467        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
1468    }
1469
1470    static void put_get_post_s_47(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2)
1471    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
1472                sa..contents = sb..contents & sa..size = sb..size"
1473        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1474        ensures "True" */
1475    {
1476        Object r1a = sa.put(k1, v1);
1477        Object r2a = sa.get(k2);
1478        /*: assume "k1 ~= k2 | r1a = v1" */
1479
1480        Object r2b = sb.get(k2);
1481        Object r1b = sb.put(k1, v1);
1482
1483        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
1484    }
1485
1486    static void put_get_post_c_47(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2)
1487    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
1488                sa..contents = sb..contents & sa..size = sb..size"
1489        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1490        ensures "True" */
1491    {
1492        Object r1a = sa.put(k1, v1);
1493        Object r2a = sa.get(k2);
1494        /*: assume "~(k1 ~= k2 | r1a = v1)" */
1495
1496        Object r2b = sb.get(k2);
1497        Object r1b = sb.put(k1, v1);
1498
1499        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
1500    }
1501
1502    static void put_put_pre_s_48(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2, Object v2)
1503    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null & v2 ~= null &
1504                sa..contents = sb..contents & sa..size = sb..size"
1505        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1506        ensures "True" */
1507    {
1508        /*: assume "k1 ~= k2 | ((k1, v1) : sa..contents & v1 = v2)" */
1509        Object r1a = sa.put(k1, v1);
1510        Object r2a = sa.put(k2, v2);
1511
1512        Object r2b = sb.put(k2, v2);
1513        Object r1b = sb.put(k1, v1);
1514
1515        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
1516    }
```

```
1517
1518    static void put_put_pre_c_48(AssociationList sa, AssociationList sb, Object k1,
            Object v1, Object k2, Object v2)
1519    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
            null & v2 ~= null &
1520                sa..contents = sb..contents & sa..size = sb..size"
1521        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1522        ensures "True" */
1523    {
1524        /*: assume "~(k1 ~= k2 | ((k1, v1) : sa..contents & v1 = v2))" */
1525        Object r1a = sa.put(k1, v1);
1526        Object r2a = sa.put(k2, v2);
1527
1528        Object r2b = sb.put(k2, v2);
1529        Object r1b = sb.put(k1, v1);
1530
1531        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
1532    }
1533
1534    static void put_put_between_s_49(AssociationList sa, AssociationList sb, Object k1,
            Object v1, Object k2, Object v2)
1535    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
            null & v2 ~= null &
1536                sa..contents = sb..contents & sa..size = sb..size"
1537        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1538        ensures "True" */
1539    {
1540        Object r1a = sa.put(k1, v1);
1541        /*: assume "k1 ~= k2 | (r1a = v1 & v1 = v2)" */
1542        Object r2a = sa.put(k2, v2);
1543
1544        Object r2b = sb.put(k2, v2);
1545        Object r1b = sb.put(k1, v1);
1546
1547        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
1548    }
1549
1550    static void put_put_between_c_49(AssociationList sa, AssociationList sb, Object k1,
            Object v1, Object k2, Object v2)
1551    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
            null & v2 ~= null &
1552                sa..contents = sb..contents & sa..size = sb..size"
1553        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1554        ensures "True" */
1555    {
1556        Object r1a = sa.put(k1, v1);
1557        /*: assume "~(k1 ~= k2 | (r1a = v1 & v1 = v2))" */
1558        Object r2a = sa.put(k2, v2);
1559
1560        Object r2b = sb.put(k2, v2);
1561        Object r1b = sb.put(k1, v1);
1562
1563        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
1564    }
1565
1566    static void put_put_post_s_50(AssociationList sa, AssociationList sb, Object k1,
            Object v1, Object k2, Object v2)
1567    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
            null & v2 ~= null &
1568                sa..contents = sb..contents & sa..size = sb..size"
1569        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1570        ensures "True" */
```

```
1571    {
1572        Object r1a = sa.put(k1, v1);
1573        Object r2a = sa.put(k2, v2);
1574        /*: assume "k1 ~= k2 | (r1a = v1 & v1 = v2)" */
1575
1576        Object r2b = sb.put(k2, v2);
1577        Object r1b = sb.put(k1, v1);
1578
1579        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
1580    }
1581
1582    static void put_put_post_c_50(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2, Object v2)
1583    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null & v2 ~= null &
1584                sa..contents = sb..contents & sa..size = sb..size"
1585        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1586        ensures "True" */
1587    {
1588        Object r1a = sa.put(k1, v1);
1589        Object r2a = sa.put(k2, v2);
1590        /*: assume "~(k1 ~= k2 | (r1a = v1 & v1 = v2))" */
1591
1592        Object r2b = sb.put(k2, v2);
1593        Object r1b = sb.put(k1, v1);
1594
1595        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
1596    }
1597
1598    static void put_put_pre_s_51(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2, Object v2)
1599    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null & v2 ~= null &
1600                sa..contents = sb..contents & sa..size = sb..size"
1601        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1602        ensures "True" */
1603    {
1604        /*: assume "k1 ~= k2 | ((k1, v1) : sa..contents & v1 = v2)" */
1605        Object r1a = sa.put(k1, v1);
1606        sa.put(k2, v2);
1607
1608        sb.put(k2, v2);
1609        Object r1b = sb.put(k1, v1);
1610
1611        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1612    }
1613
1614    static void put_put_pre_c_51(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2, Object v2)
1615    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null & v2 ~= null &
1616                sa..contents = sb..contents & sa..size = sb..size"
1617        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1618        ensures "True" */
1619    {
1620        /*: assume "~(k1 ~= k2 | ((k1, v1) : sa..contents & v1 = v2))" */
1621        Object r1a = sa.put(k1, v1);
1622        sa.put(k2, v2);
1623
1624        sb.put(k2, v2);
1625        Object r1b = sb.put(k1, v1);
1626
1627        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
```

```
1628    }
1629
1630    static void put_put_between_s_52(AssociationList sa, AssociationList sb, Object k1,
            Object v1, Object k2, Object v2)
1631    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
            null & v2 ~= null &
1632                sa..contents = sb..contents & sa..size = sb..size"
1633        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1634        ensures "True" */
1635    {
1636        Object r1a = sa.put(k1, v1);
1637        /*: assume "k1 ~= k2 | (r1a = v1 & v1 = v2)" */
1638        sa.put(k2, v2);
1639
1640        sb.put(k2, v2);
1641        Object r1b = sb.put(k1, v1);
1642
1643        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1644    }
1645
1646    static void put_put_between_c_52(AssociationList sa, AssociationList sb, Object k1,
            Object v1, Object k2, Object v2)
1647    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
            null & v2 ~= null &
1648                sa..contents = sb..contents & sa..size = sb..size"
1649        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1650        ensures "True" */
1651    {
1652        Object r1a = sa.put(k1, v1);
1653        /*: assume "~(k1 ~= k2 | (r1a = v1 & v1 = v2))" */
1654        sa.put(k2, v2);
1655
1656        sb.put(k2, v2);
1657        Object r1b = sb.put(k1, v1);
1658
1659        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1660    }
1661
1662    static void put_put_post_s_53(AssociationList sa, AssociationList sb, Object k1,
            Object v1, Object k2, Object v2)
1663    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
            null & v2 ~= null &
1664                sa..contents = sb..contents & sa..size = sb..size"
1665        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1666        ensures "True" */
1667    {
1668        Object r1a = sa.put(k1, v1);
1669        sa.put(k2, v2);
1670        /*: assume "k1 ~= k2 | (r1a = v1 & v1 = v2)" */
1671
1672        sb.put(k2, v2);
1673        Object r1b = sb.put(k1, v1);
1674
1675        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1676    }
1677
1678    static void put_put_post_c_53(AssociationList sa, AssociationList sb, Object k1,
            Object v1, Object k2, Object v2)
1679    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
            null & v2 ~= null &
1680                sa..contents = sb..contents & sa..size = sb..size"
1681        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1682        ensures "True" */
1683    {
1684        Object r1a = sa.put(k1, v1);
```

```
1685        sa.put(k2, v2);
1686        /*: assume "~(k1 ~= k2 | (r1a = v1 & v1 = v2))" */
1687
1688        sb.put(k2, v2);
1689        Object r1b = sb.put(k1, v1);
1690
1691        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1692    }
1693
1694    static void put_remove_pre_s_54(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2)
1695    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
1696                sa..contents = sb..contents & sa..size = sb..size"
1697        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1698        ensures "True" */
1699    {
1700        /*: assume "k1 ~= k2" */
1701        Object r1a = sa.put(k1, v1);
1702        Object r2a = sa.remove(k2);
1703
1704        Object r2b = sb.remove(k2);
1705        Object r1b = sb.put(k1, v1);
1706
1707        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
1708    }
1709
1710    static void put_remove_pre_c_54(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2)
1711    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
1712                sa..contents = sb..contents & sa..size = sb..size"
1713        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1714        ensures "True" */
1715    {
1716        /*: assume "~(k1 ~= k2)" */
1717        Object r1a = sa.put(k1, v1);
1718        Object r2a = sa.remove(k2);
1719
1720        Object r2b = sb.remove(k2);
1721        Object r1b = sb.put(k1, v1);
1722
1723        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
1724    }
1725
1726    static void put_remove_between_s_55(AssociationList sa, AssociationList sb, Object
        k1, Object v1, Object k2)
1727    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
1728                sa..contents = sb..contents & sa..size = sb..size"
1729        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1730        ensures "True" */
1731    {
1732        Object r1a = sa.put(k1, v1);
1733        /*: assume "k1 ~= k2" */
1734        Object r2a = sa.remove(k2);
1735
1736        Object r2b = sb.remove(k2);
1737        Object r1b = sb.put(k1, v1);
1738
1739        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
1740    }
```

```
1741
1742    static void put_remove_between_c_55(AssociationList sa, AssociationList sb, Object
            k1, Object v1, Object k2)
1743    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
            null &
1744                sa..contents = sb..contents & sa..size = sb..size"
1745        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1746        ensures "True" */
1747    {
1748        Object r1a = sa.put(k1, v1);
1749        /*: assume "~(k1 ~= k2)" */
1750        Object r2a = sa.remove(k2);
1751
1752        Object r2b = sb.remove(k2);
1753        Object r1b = sb.put(k1, v1);
1754
1755        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
1756    }
1757
1758    static void put_remove_post_s_56(AssociationList sa, AssociationList sb, Object k1,
            Object v1, Object k2)
1759    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
            null &
1760                sa..contents = sb..contents & sa..size = sb..size"
1761        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1762        ensures "True" */
1763    {
1764        Object r1a = sa.put(k1, v1);
1765        Object r2a = sa.remove(k2);
1766        /*: assume "k1 ~= k2" */
1767
1768        Object r2b = sb.remove(k2);
1769        Object r1b = sb.put(k1, v1);
1770
1771        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
1772    }
1773
1774    static void put_remove_post_c_56(AssociationList sa, AssociationList sb, Object k1,
            Object v1, Object k2)
1775    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
            null &
1776                sa..contents = sb..contents & sa..size = sb..size"
1777        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1778        ensures "True" */
1779    {
1780        Object r1a = sa.put(k1, v1);
1781        Object r2a = sa.remove(k2);
1782        /*: assume "~(k1 ~= k2)" */
1783
1784        Object r2b = sb.remove(k2);
1785        Object r1b = sb.put(k1, v1);
1786
1787        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
1788    }
1789
1790    static void put_remove_pre_s_57(AssociationList sa, AssociationList sb, Object k1,
            Object v1, Object k2)
1791    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
            null &
1792                sa..contents = sb..contents & sa..size = sb..size"
1793        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1794        ensures "True" */
```

```
1795    {
1796        /*: assume "k1 ~= k2" */
1797        Object r1a = sa.put(k1, v1);
1798        sa.remove(k2);
1799
1800        sb.remove(k2);
1801        Object r1b = sb.put(k1, v1);
1802
1803        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1804    }
1805
1806    static void put_remove_pre_c_57(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2)
1807    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
1808                sa..contents = sb..contents & sa..size = sb..size"
1809        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1810        ensures "True" */
1811    {
1812        /*: assume "~(k1 ~= k2)" */
1813        Object r1a = sa.put(k1, v1);
1814        sa.remove(k2);
1815
1816        sb.remove(k2);
1817        Object r1b = sb.put(k1, v1);
1818
1819        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1820    }
1821
1822    static void put_remove_between_s_58(AssociationList sa, AssociationList sb, Object
        k1, Object v1, Object k2)
1823    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
1824                sa..contents = sb..contents & sa..size = sb..size"
1825        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1826        ensures "True" */
1827    {
1828        Object r1a = sa.put(k1, v1);
1829        /*: assume "k1 ~= k2" */
1830        sa.remove(k2);
1831
1832        sb.remove(k2);
1833        Object r1b = sb.put(k1, v1);
1834
1835        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1836    }
1837
1838    static void put_remove_between_c_58(AssociationList sa, AssociationList sb, Object
        k1, Object v1, Object k2)
1839    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
1840                sa..contents = sb..contents & sa..size = sb..size"
1841        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1842        ensures "True" */
1843    {
1844        Object r1a = sa.put(k1, v1);
1845        /*: assume "~(k1 ~= k2)" */
1846        sa.remove(k2);
1847
1848        sb.remove(k2);
1849        Object r1b = sb.put(k1, v1);
1850
1851        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1852    }
1853
```

```
    static void put_remove_post_s_59(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.put(k1, v1);
        sa.remove(k2);
        /*: assume "k1 ~= k2" */

        sb.remove(k2);
        Object r1b = sb.put(k1, v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void put_remove_post_c_59(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.put(k1, v1);
        sa.remove(k2);
        /*: assume "~(k1 ~= k2)" */

        sb.remove(k2);
        Object r1b = sb.put(k1, v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void put_size_pre_s_60(AssociationList sa, AssociationList sb, Object k1,
        Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "EX v. (k1, v) : sa..contents" */
        Object r1a = sa.put(k1, v1);
        int r2a = sa.size();

        int r2b = sb.size();
        Object r1b = sb.put(k1, v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void put_size_pre_c_60(AssociationList sa, AssociationList sb, Object k1,
        Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(EX v. (k1, v) : sa..contents)" */
        Object r1a = sa.put(k1, v1);
        int r2a = sa.size();

```

```
        int r2b = sb.size();
        Object r1b = sb.put(k1, v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void put_size_between_s_61(AssociationList sa, AssociationList sb, Object
        k1, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.put(k1, v1);
        /*: assume "r1a ~= null" */
        int r2a = sa.size();

        int r2b = sb.size();
        Object r1b = sb.put(k1, v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void put_size_between_c_61(AssociationList sa, AssociationList sb, Object
        k1, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.put(k1, v1);
        /*: assume "~(r1a ~= null)" */
        int r2a = sa.size();

        int r2b = sb.size();
        Object r1b = sb.put(k1, v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void put_size_post_s_62(AssociationList sa, AssociationList sb, Object k1,
        Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.put(k1, v1);
        int r2a = sa.size();
        /*: assume "r1a ~= null" */

        int r2b = sb.size();
        Object r1b = sb.put(k1, v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void put_size_post_c_62(AssociationList sa, AssociationList sb, Object k1,
        Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
```

```
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.put(k1, v1);
        int r2a = sa.size();
        /*: assume "~(r1a ~= null)" */

        int r2b = sb.size();
        Object r1b = sb.put(k1, v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void put_containsKey_pre_s_63(AssociationList sa, AssociationList sb, Object
        k1, Object v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "k1 ~= k2 | (EX v. (k1, v) : sa..contents)" */
        sa.put(k1, v1);
        boolean r2a = sa.containsKey(k2);

        boolean r2b = sb.containsKey(k2);
        sb.put(k1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void put_containsKey_pre_c_63(AssociationList sa, AssociationList sb, Object
        k1, Object v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(k1 ~= k2 | (EX v. (k1, v) : sa..contents))" */
        sa.put(k1, v1);
        boolean r2a = sa.containsKey(k2);

        boolean r2b = sb.containsKey(k2);
        sb.put(k1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void put_containsKey_between_s_64(AssociationList sa, AssociationList sb,
        Object k1, Object v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.put(k1, v1);
        /*: assume "k1 ~= k2 | (EX v. (k1, v) : sa..(old contents))" */
        boolean r2a = sa.containsKey(k2);

        boolean r2b = sb.containsKey(k2);
        sb.put(k1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }
```

```
    static void put_containsKey_between_c_64(AssociationList sa, AssociationList sb,
        Object k1, Object v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.put(k1, v1);
        /*: assume "~(k1 ~= k2 | (EX v. (k1, v) : sa..(old contents)))" */
        boolean r2a = sa.containsKey(k2);

        boolean r2b = sb.containsKey(k2);
        sb.put(k1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void put_containsKey_post_s_65(AssociationList sa, AssociationList sb,
        Object k1, Object v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.put(k1, v1);
        boolean r2a = sa.containsKey(k2);
        /*: assume "k1 ~= k2 | (EX v. (k1, v) : sa..(old contents))" */

        boolean r2b = sb.containsKey(k2);
        sb.put(k1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void put_containsKey_post_c_65(AssociationList sa, AssociationList sb,
        Object k1, Object v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.put(k1, v1);
        boolean r2a = sa.containsKey(k2);
        /*: assume "~(k1 ~= k2 | (EX v. (k1, v) : sa..(old contents)))" */

        boolean r2b = sb.containsKey(k2);
        sb.put(k1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void put_get_pre_s_66(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "k1 ~= k2 | (k1, v1) : sa..contents" */
        sa.put(k1, v1);
        Object r2a = sa.get(k2);

        Object r2b = sb.get(k2);
        sb.put(k1, v1);
```

```
        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void put_get_pre_c_66(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(k1 ~= k2 | (k1, v1) : sa..contents)" */
        sa.put(k1, v1);
        Object r2a = sa.get(k2);

        Object r2b = sb.get(k2);
        sb.put(k1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void put_get_between_s_67(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.put(k1, v1);
        /*: assume "k1 ~= k2 | (k1, v1) : sa..(old contents)" */
        Object r2a = sa.get(k2);

        Object r2b = sb.get(k2);
        sb.put(k1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void put_get_between_c_67(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.put(k1, v1);
        /*: assume "~(k1 ~= k2 | (k1, v1) : sa..(old contents))" */
        Object r2a = sa.get(k2);

        Object r2b = sb.get(k2);
        sb.put(k1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void put_get_post_s_68(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
```

```
2147    {
2148        sa.put(k1, v1);
2149        Object r2a = sa.get(k2);
2150        /*: assume "k1 ~= k2 | (k1, v1) : sa..(old contents)" */
2151
2152        Object r2b = sb.get(k2);
2153        sb.put(k1, v1);
2154
2155        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2156    }
2157
2158    static void put_get_post_c_68(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2)
2159    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
2160                sa..contents = sb..contents & sa..size = sb..size"
2161        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2162        ensures "True" */
2163    {
2164        sa.put(k1, v1);
2165        Object r2a = sa.get(k2);
2166        /*: assume "~(k1 ~= k2 | (k1, v1) : sa..(old contents))" */
2167
2168        Object r2b = sb.get(k2);
2169        sb.put(k1, v1);
2170
2171        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2172    }
2173
2174    static void put_put_pre_s_69(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2, Object v2)
2175    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null & v2 ~= null &
2176                sa..contents = sb..contents & sa..size = sb..size"
2177        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2178        ensures "True" */
2179    {
2180        /*: assume "k1 ~= k2 | ((k1, v1) : sa..contents & v1 = v2)" */
2181        sa.put(k1, v1);
2182        Object r2a = sa.put(k2, v2);
2183
2184        Object r2b = sb.put(k2, v2);
2185        sb.put(k1, v1);
2186
2187        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2188    }
2189
2190    static void put_put_pre_c_69(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2, Object v2)
2191    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null & v2 ~= null &
2192                sa..contents = sb..contents & sa..size = sb..size"
2193        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2194        ensures "True" */
2195    {
2196        /*: assume "~(k1 ~= k2 | ((k1, v1) : sa..contents & v1 = v2))" */
2197        sa.put(k1, v1);
2198        Object r2a = sa.put(k2, v2);
2199
2200        Object r2b = sb.put(k2, v2);
2201        sb.put(k1, v1);
2202
2203        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2204    }
2205
```

```
2206    static void put_put_between_s_70(AssociationList sa, AssociationList sb, Object k1,
            Object v1, Object k2, Object v2)
2207    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
            null & v2 ~= null &
2208                    sa..contents = sb..contents & sa..size = sb..size"
2209        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2210        ensures "True" */
2211    {
2212        sa.put(k1, v1);
2213        /*: assume "k1 ~= k2 | ((k1, v1) : sa..(old contents) & v1 = v2)" */
2214        Object r2a = sa.put(k2, v2);
2215
2216        Object r2b = sb.put(k2, v2);
2217        sb.put(k1, v1);
2218
2219        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2220    }
2221
2222    static void put_put_between_c_70(AssociationList sa, AssociationList sb, Object k1,
            Object v1, Object k2, Object v2)
2223    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
            null & v2 ~= null &
2224                    sa..contents = sb..contents & sa..size = sb..size"
2225        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2226        ensures "True" */
2227    {
2228        sa.put(k1, v1);
2229        /*: assume "~(k1 ~= k2 | ((k1, v1) : sa..(old contents) & v1 = v2))" */
2230        Object r2a = sa.put(k2, v2);
2231
2232        Object r2b = sb.put(k2, v2);
2233        sb.put(k1, v1);
2234
2235        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2236    }
2237
2238    static void put_put_post_s_71(AssociationList sa, AssociationList sb, Object k1,
            Object v1, Object k2, Object v2)
2239    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
            null & v2 ~= null &
2240                    sa..contents = sb..contents & sa..size = sb..size"
2241        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2242        ensures "True" */
2243    {
2244        sa.put(k1, v1);
2245        Object r2a = sa.put(k2, v2);
2246        /*: assume "k1 ~= k2 | ((k1, v1) : sa..(old contents) & v1 = v2)" */
2247
2248        Object r2b = sb.put(k2, v2);
2249        sb.put(k1, v1);
2250
2251        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2252    }
2253
2254    static void put_put_post_c_71(AssociationList sa, AssociationList sb, Object k1,
            Object v1, Object k2, Object v2)
2255    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
            null & v2 ~= null &
2256                    sa..contents = sb..contents & sa..size = sb..size"
2257        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2258        ensures "True" */
2259    {
2260        sa.put(k1, v1);
2261        Object r2a = sa.put(k2, v2);
2262        /*: assume "~(k1 ~= k2 | ((k1, v1) : sa..(old contents) & v1 = v2))" */
```

```
2263
2264            Object r2b = sb.put(k2, v2);
2265            sb.put(k1, v1);
2266
2267            /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2268        }
2269
2270        static void put_put_pre_s_72(AssociationList sa, AssociationList sb, Object k1,
                Object v1, Object k2, Object v2)
2271        /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
                null & v2 ~= null &
2272                        sa..contents = sb..contents & sa..size = sb..size"
2273            modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2274            ensures "True" */
2275        {
2276            /*: assume "k1 ~= k2 | v1 = v2" */
2277            sa.put(k1, v1);
2278            sa.put(k2, v2);
2279
2280            sb.put(k2, v2);
2281            sb.put(k1, v1);
2282
2283            /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
2284        }
2285
2286        static void put_put_pre_c_72(AssociationList sa, AssociationList sb, Object k1,
                Object v1, Object k2, Object v2)
2287        /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
                null & v2 ~= null &
2288                        sa..contents = sb..contents & sa..size = sb..size"
2289            modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2290            ensures "True" */
2291        {
2292            /*: assume "~(k1 ~= k2 | v1 = v2)" */
2293            sa.put(k1, v1);
2294            sa.put(k2, v2);
2295
2296            sb.put(k2, v2);
2297            sb.put(k1, v1);
2298
2299            /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
2300        }
2301
2302        static void put_put_between_s_73(AssociationList sa, AssociationList sb, Object k1,
                Object v1, Object k2, Object v2)
2303        /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
                null & v2 ~= null &
2304                        sa..contents = sb..contents & sa..size = sb..size"
2305            modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2306            ensures "True" */
2307        {
2308            sa.put(k1, v1);
2309            /*: assume "k1 ~= k2 | v1 = v2" */
2310            sa.put(k2, v2);
2311
2312            sb.put(k2, v2);
2313            sb.put(k1, v1);
2314
2315            /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
2316        }
2317
2318        static void put_put_between_c_73(AssociationList sa, AssociationList sb, Object k1,
                Object v1, Object k2, Object v2)
2319        /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
                null & v2 ~= null &
```

```
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.put(k1, v1);
        /*: assume "~(k1 ~= k2 | v1 = v2)" */
        sa.put(k2, v2);

        sb.put(k2, v2);
        sb.put(k1, v1);

        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void put_put_post_s_74(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null & v2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.put(k1, v1);
        sa.put(k2, v2);
        /*: assume "k1 ~= k2 | v1 = v2" */

        sb.put(k2, v2);
        sb.put(k1, v1);

        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void put_put_post_c_74(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null & v2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.put(k1, v1);
        sa.put(k2, v2);
        /*: assume "~(k1 ~= k2 | v1 = v2)" */

        sb.put(k2, v2);
        sb.put(k1, v1);

        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void put_remove_pre_s_75(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "k1 ~= k2" */
        sa.put(k1, v1);
        Object r2a = sa.remove(k2);

        Object r2b = sb.remove(k2);
        sb.put(k1, v1);

```

```
2379        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2380    }
2381
2382    static void put_remove_pre_c_75(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2)
2383    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
2384                sa..contents = sb..contents & sa..size = sb..size"
2385        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2386        ensures "True" */
2387    {
2388        /*: assume "~(k1 ~= k2)" */
2389        sa.put(k1, v1);
2390        Object r2a = sa.remove(k2);
2391
2392        Object r2b = sb.remove(k2);
2393        sb.put(k1, v1);
2394
2395        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2396    }
2397
2398    static void put_remove_between_s_76(AssociationList sa, AssociationList sb, Object
        k1, Object v1, Object k2)
2399    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
2400                sa..contents = sb..contents & sa..size = sb..size"
2401        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2402        ensures "True" */
2403    {
2404        sa.put(k1, v1);
2405        /*: assume "k1 ~= k2" */
2406        Object r2a = sa.remove(k2);
2407
2408        Object r2b = sb.remove(k2);
2409        sb.put(k1, v1);
2410
2411        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2412    }
2413
2414    static void put_remove_between_c_76(AssociationList sa, AssociationList sb, Object
        k1, Object v1, Object k2)
2415    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
2416                sa..contents = sb..contents & sa..size = sb..size"
2417        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2418        ensures "True" */
2419    {
2420        sa.put(k1, v1);
2421        /*: assume "~(k1 ~= k2)" */
2422        Object r2a = sa.remove(k2);
2423
2424        Object r2b = sb.remove(k2);
2425        sb.put(k1, v1);
2426
2427        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2428    }
2429
2430    static void put_remove_post_s_77(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2)
2431    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
2432                sa..contents = sb..contents & sa..size = sb..size"
2433        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2434        ensures "True" */
2435    {
```

```
2436        sa.put(k1, v1);
2437        Object r2a = sa.remove(k2);
2438        /*: assume "k1 ~= k2" */
2439
2440        Object r2b = sb.remove(k2);
2441        sb.put(k1, v1);
2442
2443        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2444    }
2445
2446    static void put_remove_post_c_77(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2)
2447    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
2448                sa..contents = sb..contents & sa..size = sb..size"
2449        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2450        ensures "True" */
2451    {
2452        sa.put(k1, v1);
2453        Object r2a = sa.remove(k2);
2454        /*: assume "~(k1 ~= k2)" */
2455
2456        Object r2b = sb.remove(k2);
2457        sb.put(k1, v1);
2458
2459        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2460    }
2461
2462    static void put_remove_pre_s_78(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2)
2463    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
2464                sa..contents = sb..contents & sa..size = sb..size"
2465        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2466        ensures "True" */
2467    {
2468        /*: assume "k1 ~= k2" */
2469        sa.put(k1, v1);
2470        sa.remove(k2);
2471
2472        sb.remove(k2);
2473        sb.put(k1, v1);
2474
2475        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
2476    }
2477
2478    static void put_remove_pre_c_78(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2)
2479    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
2480                sa..contents = sb..contents & sa..size = sb..size"
2481        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2482        ensures "True" */
2483    {
2484        /*: assume "~(k1 ~= k2)" */
2485        sa.put(k1, v1);
2486        sa.remove(k2);
2487
2488        sb.remove(k2);
2489        sb.put(k1, v1);
2490
2491        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
2492    }
2493
```

```
    static void put_remove_between_s_79(AssociationList sa, AssociationList sb, Object
        k1, Object v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.put(k1, v1);
        /*: assume "k1 ~= k2" */
        sa.remove(k2);

        sb.remove(k2);
        sb.put(k1, v1);

        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void put_remove_between_c_79(AssociationList sa, AssociationList sb, Object
        k1, Object v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.put(k1, v1);
        /*: assume "~(k1 ~= k2)" */
        sa.remove(k2);

        sb.remove(k2);
        sb.put(k1, v1);

        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void put_remove_post_s_80(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.put(k1, v1);
        sa.remove(k2);
        /*: assume "k1 ~= k2" */

        sb.remove(k2);
        sb.put(k1, v1);

        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void put_remove_post_c_80(AssociationList sa, AssociationList sb, Object k1,
        Object v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
        null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.put(k1, v1);
        sa.remove(k2);
        /*: assume "~(k1 ~= k2)" */
```

```
2551
2552            sb.remove(k2);
2553            sb.put(k1, v1);
2554
2555            /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
2556        }
2557
2558        static void put_size_pre_s_81(AssociationList sa, AssociationList sb, Object k1,
                    Object v1)
2559        /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
2560                      sa..contents = sb..contents & sa..size = sb..size"
2561            modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2562            ensures "True" */
2563        {
2564            /*: assume "EX v. (k1, v) : sa..contents" */
2565            sa.put(k1, v1);
2566            int r2a = sa.size();
2567
2568            int r2b = sb.size();
2569            sb.put(k1, v1);
2570
2571            /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2572        }
2573
2574        static void put_size_pre_c_81(AssociationList sa, AssociationList sb, Object k1,
                    Object v1)
2575        /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
2576                      sa..contents = sb..contents & sa..size = sb..size"
2577            modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2578            ensures "True" */
2579        {
2580            /*: assume "~(EX v. (k1, v) : sa..contents)" */
2581            sa.put(k1, v1);
2582            int r2a = sa.size();
2583
2584            int r2b = sb.size();
2585            sb.put(k1, v1);
2586
2587            /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2588        }
2589
2590        static void put_size_between_s_82(AssociationList sa, AssociationList sb, Object
                    k1, Object v1)
2591        /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
2592                      sa..contents = sb..contents & sa..size = sb..size"
2593            modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2594            ensures "True" */
2595        {
2596            sa.put(k1, v1);
2597            /*: assume "EX v. (k1, v) : sa..(old contents)" */
2598            int r2a = sa.size();
2599
2600            int r2b = sb.size();
2601            sb.put(k1, v1);
2602
2603            /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2604        }
2605
2606        static void put_size_between_c_82(AssociationList sa, AssociationList sb, Object
                    k1, Object v1)
2607        /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
2608                      sa..contents = sb..contents & sa..size = sb..size"
2609            modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2610            ensures "True" */
2611        {
```

```
2612        sa.put(k1, v1);
2613        /*: assume "~(EX v. (k1, v) : sa..(old contents))" */
2614        int r2a = sa.size();
2615
2616        int r2b = sb.size();
2617        sb.put(k1, v1);
2618
2619        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2620    }
2621
2622    static void put_size_post_s_83(AssociationList sa, AssociationList sb, Object k1,
        Object v1)
2623    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
2624                  sa..contents = sb..contents & sa..size = sb..size"
2625        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2626        ensures "True" */
2627    {
2628        sa.put(k1, v1);
2629        int r2a = sa.size();
2630        /*: assume "EX v. (k1, v) : sa..(old contents)" */
2631
2632        int r2b = sb.size();
2633        sb.put(k1, v1);
2634
2635        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2636    }
2637
2638    static void put_size_post_c_83(AssociationList sa, AssociationList sb, Object k1,
        Object v1)
2639    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
2640                  sa..contents = sb..contents & sa..size = sb..size"
2641        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2642        ensures "True" */
2643    {
2644        sa.put(k1, v1);
2645        int r2a = sa.size();
2646        /*: assume "~(EX v. (k1, v) : sa..(old contents))" */
2647
2648        int r2b = sb.size();
2649        sb.put(k1, v1);
2650
2651        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2652    }
2653
2654    static void remove_containsKey_pre_s_84(AssociationList sa, AssociationList sb,
        Object k1, Object k2)
2655    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
2656                  sa..contents = sb..contents & sa..size = sb..size"
2657        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2658        ensures "True" */
2659    {
2660        /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
2661        Object r1a = sa.remove(k1);
2662        boolean r2a = sa.containsKey(k2);
2663
2664        boolean r2b = sb.containsKey(k2);
2665        Object r1b = sb.remove(k1);
2666
2667        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
2668    }
2669
2670    static void remove_containsKey_pre_c_84(AssociationList sa, AssociationList sb,
        Object k1, Object k2)
2671    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
```

```
2672                 sa..contents = sb..contents & sa..size = sb..size"
2673         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2674         ensures "True" */
2675     {
2676         /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
2677         Object r1a = sa.remove(k1);
2678         boolean r2a = sa.containsKey(k2);
2679 
2680         boolean r2b = sb.containsKey(k2);
2681         Object r1b = sb.remove(k1);
2682 
2683         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
2684     }
2685 
2686     static void remove_containsKey_between_s_85(AssociationList sa, AssociationList sb,
            Object k1, Object k2)
2687     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
2688                 sa..contents = sb..contents & sa..size = sb..size"
2689         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2690         ensures "True" */
2691     {
2692         Object r1a = sa.remove(k1);
2693         /*: assume "k1 ~= k2 | r1a = null" */
2694         boolean r2a = sa.containsKey(k2);
2695 
2696         boolean r2b = sb.containsKey(k2);
2697         Object r1b = sb.remove(k1);
2698 
2699         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
2700     }
2701 
2702     static void remove_containsKey_between_c_85(AssociationList sa, AssociationList sb,
            Object k1, Object k2)
2703     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
2704                 sa..contents = sb..contents & sa..size = sb..size"
2705         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2706         ensures "True" */
2707     {
2708         Object r1a = sa.remove(k1);
2709         /*: assume "~(k1 ~= k2 | r1a = null)" */
2710         boolean r2a = sa.containsKey(k2);
2711 
2712         boolean r2b = sb.containsKey(k2);
2713         Object r1b = sb.remove(k1);
2714 
2715         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
2716     }
2717 
2718     static void remove_containsKey_post_s_86(AssociationList sa, AssociationList sb,
            Object k1, Object k2)
2719     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
2720                 sa..contents = sb..contents & sa..size = sb..size"
2721         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2722         ensures "True" */
2723     {
2724         Object r1a = sa.remove(k1);
2725         boolean r2a = sa.containsKey(k2);
2726         /*: assume "k1 ~= k2 | r1a = null" */
2727 
2728         boolean r2b = sb.containsKey(k2);
2729         Object r1b = sb.remove(k1);
2730 
```

```
2731        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
2732    }
2733
2734    static void remove_containsKey_post_c_86(AssociationList sa, AssociationList sb,
            Object k1, Object k2)
2735    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
2736                  sa..contents = sb..contents & sa..size = sb..size"
2737        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2738        ensures "True" */
2739    {
2740        Object r1a = sa.remove(k1);
2741        boolean r2a = sa.containsKey(k2);
2742        /*: assume "~(k1 ~= k2 | r1a = null)" */
2743
2744        boolean r2b = sb.containsKey(k2);
2745        Object r1b = sb.remove(k1);
2746
2747        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
2748    }
2749
2750    static void remove_get_pre_s_87(AssociationList sa, AssociationList sb, Object k1,
            Object k2)
2751    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
2752                  sa..contents = sb..contents & sa..size = sb..size"
2753        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2754        ensures "True" */
2755    {
2756        /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
2757        Object r1a = sa.remove(k1);
2758        Object r2a = sa.get(k2);
2759
2760        Object r2b = sb.get(k2);
2761        Object r1b = sb.remove(k1);
2762
2763        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
2764    }
2765
2766    static void remove_get_pre_c_87(AssociationList sa, AssociationList sb, Object k1,
            Object k2)
2767    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
2768                  sa..contents = sb..contents & sa..size = sb..size"
2769        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2770        ensures "True" */
2771    {
2772        /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
2773        Object r1a = sa.remove(k1);
2774        Object r2a = sa.get(k2);
2775
2776        Object r2b = sb.get(k2);
2777        Object r1b = sb.remove(k1);
2778
2779        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
2780    }
2781
2782    static void remove_get_between_s_88(AssociationList sa, AssociationList sb, Object
            k1, Object k2)
2783    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
2784                  sa..contents = sb..contents & sa..size = sb..size"
2785        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2786        ensures "True" */
2787    {
```

```
2788        Object r1a = sa.remove(k1);
2789        /*: assume "k1 ~= k2 | r1a = null" */
2790        Object r2a = sa.get(k2);
2791
2792        Object r2b = sb.get(k2);
2793        Object r1b = sb.remove(k1);
2794
2795        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
2796    }
2797
2798    static void remove_get_between_c_88(AssociationList sa, AssociationList sb, Object
            k1, Object k2)
2799    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
2800                  sa..contents = sb..contents & sa..size = sb..size"
2801        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2802        ensures "True" */
2803    {
2804        Object r1a = sa.remove(k1);
2805        /*: assume "~(k1 ~= k2 | r1a = null)" */
2806        Object r2a = sa.get(k2);
2807
2808        Object r2b = sb.get(k2);
2809        Object r1b = sb.remove(k1);
2810
2811        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
2812    }
2813
2814    static void remove_get_post_s_89(AssociationList sa, AssociationList sb, Object k1,
            Object k2)
2815    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
2816                  sa..contents = sb..contents & sa..size = sb..size"
2817        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2818        ensures "True" */
2819    {
2820        Object r1a = sa.remove(k1);
2821        Object r2a = sa.get(k2);
2822        /*: assume "k1 ~= k2 | r1a = null" */
2823
2824        Object r2b = sb.get(k2);
2825        Object r1b = sb.remove(k1);
2826
2827        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
2828    }
2829
2830    static void remove_get_post_c_89(AssociationList sa, AssociationList sb, Object k1,
            Object k2)
2831    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
2832                  sa..contents = sb..contents & sa..size = sb..size"
2833        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2834        ensures "True" */
2835    {
2836        Object r1a = sa.remove(k1);
2837        Object r2a = sa.get(k2);
2838        /*: assume "~(k1 ~= k2 | r1a = null)" */
2839
2840        Object r2b = sb.get(k2);
2841        Object r1b = sb.remove(k1);
2842
2843        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
2844    }
2845
```

```
2846    static void remove_put_pre_s_90(AssociationList sa, AssociationList sb, Object k1,
            Object k2, Object v2)
2847    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
            null &
2848                sa..contents = sb..contents & sa..size = sb..size"
2849        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2850        ensures "True" */
2851    {
2852        /*: assume "k1 ~= k2" */
2853        Object r1a = sa.remove(k1);
2854        Object r2a = sa.put(k2, v2);
2855
2856        Object r2b = sb.put(k2, v2);
2857        Object r1b = sb.remove(k1);
2858
2859        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
2860    }
2861
2862    static void remove_put_pre_c_90(AssociationList sa, AssociationList sb, Object k1,
            Object k2, Object v2)
2863    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
            null &
2864                sa..contents = sb..contents & sa..size = sb..size"
2865        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2866        ensures "True" */
2867    {
2868        /*: assume "~(k1 ~= k2)" */
2869        Object r1a = sa.remove(k1);
2870        Object r2a = sa.put(k2, v2);
2871
2872        Object r2b = sb.put(k2, v2);
2873        Object r1b = sb.remove(k1);
2874
2875        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
2876    }
2877
2878    static void remove_put_between_s_91(AssociationList sa, AssociationList sb, Object
            k1, Object k2, Object v2)
2879    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
            null &
2880                sa..contents = sb..contents & sa..size = sb..size"
2881        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2882        ensures "True" */
2883    {
2884        Object r1a = sa.remove(k1);
2885        /*: assume "k1 ~= k2" */
2886        Object r2a = sa.put(k2, v2);
2887
2888        Object r2b = sb.put(k2, v2);
2889        Object r1b = sb.remove(k1);
2890
2891        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
2892    }
2893
2894    static void remove_put_between_c_91(AssociationList sa, AssociationList sb, Object
            k1, Object k2, Object v2)
2895    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
            null &
2896                sa..contents = sb..contents & sa..size = sb..size"
2897        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2898        ensures "True" */
2899    {
```

```
        Object r1a = sa.remove(k1);
        /*: assume "~(k1 ~= k2)" */
        Object r2a = sa.put(k2, v2);

        Object r2b = sb.put(k2, v2);
        Object r1b = sb.remove(k1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void remove_put_post_s_92(AssociationList sa, AssociationList sb, Object k1,
        Object k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
        null &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.remove(k1);
        Object r2a = sa.put(k2, v2);
        /*: assume "k1 ~= k2" */

        Object r2b = sb.put(k2, v2);
        Object r1b = sb.remove(k1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void remove_put_post_c_92(AssociationList sa, AssociationList sb, Object k1,
        Object k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
        null &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.remove(k1);
        Object r2a = sa.put(k2, v2);
        /*: assume "~(k1 ~= k2)" */

        Object r2b = sb.put(k2, v2);
        Object r1b = sb.remove(k1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void remove_put_pre_s_93(AssociationList sa, AssociationList sb, Object k1,
        Object k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
        null &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "k1 ~= k2" */
        Object r1a = sa.remove(k1);
        sa.put(k2, v2);

        sb.put(k2, v2);
        Object r1b = sb.remove(k1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
```

```
2956    }
2957
2958    static void remove_put_pre_c_93(AssociationList sa, AssociationList sb, Object k1,
            Object k2, Object v2)
2959    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
            null &
2960                sa..contents = sb..contents & sa..size = sb..size"
2961     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2962     ensures "True" */
2963    {
2964        /*: assume "~(k1 ~= k2)" */
2965        Object r1a = sa.remove(k1);
2966        sa.put(k2, v2);
2967
2968        sb.put(k2, v2);
2969        Object r1b = sb.remove(k1);
2970
2971        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
2972    }
2973
2974    static void remove_put_between_s_94(AssociationList sa, AssociationList sb, Object
            k1, Object k2, Object v2)
2975    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
            null &
2976                sa..contents = sb..contents & sa..size = sb..size"
2977     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2978     ensures "True" */
2979    {
2980        Object r1a = sa.remove(k1);
2981        /*: assume "k1 ~= k2" */
2982        sa.put(k2, v2);
2983
2984        sb.put(k2, v2);
2985        Object r1b = sb.remove(k1);
2986
2987        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
2988    }
2989
2990    static void remove_put_between_c_94(AssociationList sa, AssociationList sb, Object
            k1, Object k2, Object v2)
2991    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
            null &
2992                sa..contents = sb..contents & sa..size = sb..size"
2993     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2994     ensures "True" */
2995    {
2996        Object r1a = sa.remove(k1);
2997        /*: assume "~(k1 ~= k2)" */
2998        sa.put(k2, v2);
2999
3000        sb.put(k2, v2);
3001        Object r1b = sb.remove(k1);
3002
3003        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
3004    }
3005
3006    static void remove_put_post_s_95(AssociationList sa, AssociationList sb, Object k1,
            Object k2, Object v2)
3007    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
            null &
3008                sa..contents = sb..contents & sa..size = sb..size"
3009     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3010     ensures "True" */
3011    {
3012        Object r1a = sa.remove(k1);
```

```
3013        sa.put(k2, v2);
3014        /*: assume "k1 ~= k2" */
3015
3016        sb.put(k2, v2);
3017        Object r1b = sb.remove(k1);
3018
3019        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
3020    }
3021
3022    static void remove_put_post_c_95(AssociationList sa, AssociationList sb, Object k1,
        Object k2, Object v2)
3023    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
        null &
3024                sa..contents = sb..contents & sa..size = sb..size"
3025        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3026        ensures "True" */
3027    {
3028        Object r1a = sa.remove(k1);
3029        sa.put(k2, v2);
3030        /*: assume "~(k1 ~= k2)" */
3031
3032        sb.put(k2, v2);
3033        Object r1b = sb.remove(k1);
3034
3035        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
3036    }
3037
3038    static void remove_remove_pre_s_96(AssociationList sa, AssociationList sb, Object
        k1, Object k2)
3039    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3040                sa..contents = sb..contents & sa..size = sb..size"
3041        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3042        ensures "True" */
3043    {
3044        /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
3045        Object r1a = sa.remove(k1);
3046        Object r2a = sa.remove(k2);
3047
3048        Object r2b = sb.remove(k2);
3049        Object r1b = sb.remove(k1);
3050
3051        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
3052    }
3053
3054    static void remove_remove_pre_c_96(AssociationList sa, AssociationList sb, Object
        k1, Object k2)
3055    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3056                sa..contents = sb..contents & sa..size = sb..size"
3057        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3058        ensures "True" */
3059    {
3060        /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
3061        Object r1a = sa.remove(k1);
3062        Object r2a = sa.remove(k2);
3063
3064        Object r2b = sb.remove(k2);
3065        Object r1b = sb.remove(k1);
3066
3067        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
3068    }
3069
3070    static void remove_remove_between_s_97(AssociationList sa, AssociationList sb,
        Object k1, Object k2)
```

```
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.remove(k1);
        /*: assume "k1 ~= k2 | r1a = null" */
        Object r2a = sa.remove(k2);

        Object r2b = sb.remove(k2);
        Object r1b = sb.remove(k1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void remove_remove_between_c_97(AssociationList sa, AssociationList sb,
        Object k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.remove(k1);
        /*: assume "~(k1 ~= k2 | r1a = null)" */
        Object r2a = sa.remove(k2);

        Object r2b = sb.remove(k2);
        Object r1b = sb.remove(k1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void remove_remove_post_s_98(AssociationList sa, AssociationList sb, Object
        k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.remove(k1);
        Object r2a = sa.remove(k2);
        /*: assume "k1 ~= k2 | r1a = null" */

        Object r2b = sb.remove(k2);
        Object r1b = sb.remove(k1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void remove_remove_post_c_98(AssociationList sa, AssociationList sb, Object
        k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.remove(k1);
        Object r2a = sa.remove(k2);
        /*: assume "~(k1 ~= k2 | r1a = null)" */

        Object r2b = sb.remove(k2);
        Object r1b = sb.remove(k1);
```

```
3130
3131        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
3132    }
3133
3134    static void remove_remove_pre_s_99(AssociationList sa, AssociationList sb, Object
            k1, Object k2)
3135    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3136                  sa..contents = sb..contents & sa..size = sb..size"
3137        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3138        ensures "True" */
3139    {
3140        /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
3141        Object r1a = sa.remove(k1);
3142        sa.remove(k2);
3143
3144        sb.remove(k2);
3145        Object r1b = sb.remove(k1);
3146
3147        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
3148    }
3149
3150    static void remove_remove_pre_c_99(AssociationList sa, AssociationList sb, Object
            k1, Object k2)
3151    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3152                  sa..contents = sb..contents & sa..size = sb..size"
3153        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3154        ensures "True" */
3155    {
3156        /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
3157        Object r1a = sa.remove(k1);
3158        sa.remove(k2);
3159
3160        sb.remove(k2);
3161        Object r1b = sb.remove(k1);
3162
3163        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
3164    }
3165
3166    static void remove_remove_between_s_100(AssociationList sa, AssociationList sb,
            Object k1, Object k2)
3167    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3168                  sa..contents = sb..contents & sa..size = sb..size"
3169        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3170        ensures "True" */
3171    {
3172        Object r1a = sa.remove(k1);
3173        /*: assume "k1 ~= k2 | r1a = null" */
3174        sa.remove(k2);
3175
3176        sb.remove(k2);
3177        Object r1b = sb.remove(k1);
3178
3179        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
3180    }
3181
3182    static void remove_remove_between_c_100(AssociationList sa, AssociationList sb,
            Object k1, Object k2)
3183    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3184                  sa..contents = sb..contents & sa..size = sb..size"
3185        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3186        ensures "True" */
3187    {
3188        Object r1a = sa.remove(k1);
3189        /*: assume "~(k1 ~= k2 | r1a = null)" */
```

```
        sa.remove(k2);

        sb.remove(k2);
        Object r1b = sb.remove(k1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_remove_post_s_101(AssociationList sa, AssociationList sb, Object
        k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.remove(k1);
        sa.remove(k2);
        /*: assume "k1 ~= k2 | r1a = null" */

        sb.remove(k2);
        Object r1b = sb.remove(k1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_remove_post_c_101(AssociationList sa, AssociationList sb, Object
        k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.remove(k1);
        sa.remove(k2);
        /*: assume "~(k1 ~= k2 | r1a = null)" */

        sb.remove(k2);
        Object r1b = sb.remove(k1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_size_pre_s_102(AssociationList sa, AssociationList sb, Object k1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(EX v. (k1, v) : sa..contents)" */
        Object r1a = sa.remove(k1);
        int r2a = sa.size();

        int r2b = sb.size();
        Object r1b = sb.remove(k1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void remove_size_pre_c_102(AssociationList sa, AssociationList sb, Object k1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
```

```
        /*: assume "~(~(EX v. (k1, v) : sa..contents))" */
        Object r1a = sa.remove(k1);
        int r2a = sa.size();

        int r2b = sb.size();
        Object r1b = sb.remove(k1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void remove_size_between_s_103(AssociationList sa, AssociationList sb,
        Object k1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.remove(k1);
        /*: assume "r1a = null" */
        int r2a = sa.size();

        int r2b = sb.size();
        Object r1b = sb.remove(k1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void remove_size_between_c_103(AssociationList sa, AssociationList sb,
        Object k1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.remove(k1);
        /*: assume "~(r1a = null)" */
        int r2a = sa.size();

        int r2b = sb.size();
        Object r1b = sb.remove(k1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void remove_size_post_s_104(AssociationList sa, AssociationList sb, Object
        k1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.remove(k1);
        int r2a = sa.size();
        /*: assume "r1a = null" */

        int r2b = sb.size();
        Object r1b = sb.remove(k1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }
```

```
3310    static void remove_size_post_c_104(AssociationList sa, AssociationList sb, Object
            k1)
3311    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
3312                 sa..contents = sb..contents & sa..size = sb..size"
3313        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3314        ensures "True" */
3315    {
3316        Object r1a = sa.remove(k1);
3317        int r2a = sa.size();
3318        /*: assume "~(r1a = null)" */
3319
3320        int r2b = sb.size();
3321        Object r1b = sb.remove(k1);
3322
3323        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
3324    }
3325
3326    static void remove_containsKey_pre_s_105(AssociationList sa, AssociationList sb,
            Object k1, Object k2)
3327    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
3328                 sa..contents = sb..contents & sa..size = sb..size"
3329        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3330        ensures "True" */
3331    {
3332        /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
3333        sa.remove(k1);
3334        boolean r2a = sa.containsKey(k2);
3335
3336        boolean r2b = sb.containsKey(k2);
3337        sb.remove(k1);
3338
3339        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3340    }
3341
3342    static void remove_containsKey_pre_c_105(AssociationList sa, AssociationList sb,
            Object k1, Object k2)
3343    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
3344                 sa..contents = sb..contents & sa..size = sb..size"
3345        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3346        ensures "True" */
3347    {
3348        /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
3349        sa.remove(k1);
3350        boolean r2a = sa.containsKey(k2);
3351
3352        boolean r2b = sb.containsKey(k2);
3353        sb.remove(k1);
3354
3355        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3356    }
3357
3358    static void remove_containsKey_between_s_106(AssociationList sa, AssociationList
            sb, Object k1, Object k2)
3359    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
3360                 sa..contents = sb..contents & sa..size = sb..size"
3361        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3362        ensures "True" */
3363    {
3364        sa.remove(k1);
3365        /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents))" */
3366        boolean r2a = sa.containsKey(k2);
3367
3368        boolean r2b = sb.containsKey(k2);
3369        sb.remove(k1);
```

```
3370
3371        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3372    }
3373
3374    static void remove_containsKey_between_c_106(AssociationList sa, AssociationList
        sb, Object k1, Object k2)
3375    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
3376                sa..contents = sb..contents & sa..size = sb..size"
3377        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3378        ensures "True" */
3379    {
3380        sa.remove(k1);
3381        /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents)))" */
3382        boolean r2a = sa.containsKey(k2);
3383
3384        boolean r2b = sb.containsKey(k2);
3385        sb.remove(k1);
3386
3387        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3388    }
3389
3390    static void remove_containsKey_post_s_107(AssociationList sa, AssociationList sb,
        Object k1, Object k2)
3391    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
3392                sa..contents = sb..contents & sa..size = sb..size"
3393        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3394        ensures "True" */
3395    {
3396        sa.remove(k1);
3397        boolean r2a = sa.containsKey(k2);
3398        /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents))" */
3399
3400        boolean r2b = sb.containsKey(k2);
3401        sb.remove(k1);
3402
3403        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3404    }
3405
3406    static void remove_containsKey_post_c_107(AssociationList sa, AssociationList sb,
        Object k1, Object k2)
3407    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
3408                sa..contents = sb..contents & sa..size = sb..size"
3409        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3410        ensures "True" */
3411    {
3412        sa.remove(k1);
3413        boolean r2a = sa.containsKey(k2);
3414        /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents)))" */
3415
3416        boolean r2b = sb.containsKey(k2);
3417        sb.remove(k1);
3418
3419        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3420    }
3421
3422    static void remove_get_pre_s_108(AssociationList sa, AssociationList sb, Object k1,
        Object k2)
3423    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3424                sa..contents = sb..contents & sa..size = sb..size"
3425        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3426        ensures "True" */
3427    {
3428        /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
3429        sa.remove(k1);
3430        Object r2a = sa.get(k2);
```

```
3431
3432            Object r2b = sb.get(k2);
3433            sb.remove(k1);
3434
3435            /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3436        }
3437
3438        static void remove_get_pre_c_108(AssociationList sa, AssociationList sb, Object k1,
                Object k2)
3439        /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3440                      sa..contents = sb..contents & sa..size = sb..size"
3441            modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3442            ensures "True" */
3443        {
3444            /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
3445            sa.remove(k1);
3446            Object r2a = sa.get(k2);
3447
3448            Object r2b = sb.get(k2);
3449            sb.remove(k1);
3450
3451            /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3452        }
3453
3454        static void remove_get_between_s_109(AssociationList sa, AssociationList sb, Object
                k1, Object k2)
3455        /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3456                      sa..contents = sb..contents & sa..size = sb..size"
3457            modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3458            ensures "True" */
3459        {
3460            sa.remove(k1);
3461            /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents))" */
3462            Object r2a = sa.get(k2);
3463
3464            Object r2b = sb.get(k2);
3465            sb.remove(k1);
3466
3467            /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3468        }
3469
3470        static void remove_get_between_c_109(AssociationList sa, AssociationList sb, Object
                k1, Object k2)
3471        /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3472                      sa..contents = sb..contents & sa..size = sb..size"
3473            modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3474            ensures "True" */
3475        {
3476            sa.remove(k1);
3477            /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents)))" */
3478            Object r2a = sa.get(k2);
3479
3480            Object r2b = sb.get(k2);
3481            sb.remove(k1);
3482
3483            /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3484        }
3485
3486        static void remove_get_post_s_110(AssociationList sa, AssociationList sb, Object
                k1, Object k2)
3487        /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3488                      sa..contents = sb..contents & sa..size = sb..size"
3489            modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3490            ensures "True" */
3491        {
```

```
3492        sa.remove(k1);
3493        Object r2a = sa.get(k2);
3494        /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents))" */
3495
3496        Object r2b = sb.get(k2);
3497        sb.remove(k1);
3498
3499        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3500    }
3501
3502    static void remove_get_post_c_110(AssociationList sa, AssociationList sb, Object
        k1, Object k2)
3503    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3504                sa..contents = sb..contents & sa..size = sb..size"
3505        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3506        ensures "True" */
3507    {
3508        sa.remove(k1);
3509        Object r2a = sa.get(k2);
3510        /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents)))" */
3511
3512        Object r2b = sb.get(k2);
3513        sb.remove(k1);
3514
3515        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3516    }
3517
3518    static void remove_put_pre_s_111(AssociationList sa, AssociationList sb, Object k1,
        Object k2, Object v2)
3519    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
        null &
3520                sa..contents = sb..contents & sa..size = sb..size"
3521        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3522        ensures "True" */
3523    {
3524        /*: assume "k1 ~= k2" */
3525        sa.remove(k1);
3526        Object r2a = sa.put(k2, v2);
3527
3528        Object r2b = sb.put(k2, v2);
3529        sb.remove(k1);
3530
3531        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3532    }
3533
3534    static void remove_put_pre_c_111(AssociationList sa, AssociationList sb, Object k1,
        Object k2, Object v2)
3535    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
        null &
3536                sa..contents = sb..contents & sa..size = sb..size"
3537        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3538        ensures "True" */
3539    {
3540        /*: assume "~(k1 ~= k2)" */
3541        sa.remove(k1);
3542        Object r2a = sa.put(k2, v2);
3543
3544        Object r2b = sb.put(k2, v2);
3545        sb.remove(k1);
3546
3547        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3548    }
3549
3550    static void remove_put_between_s_112(AssociationList sa, AssociationList sb, Object
        k1, Object k2, Object v2)
```

```
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
        null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(k1);
        /*: assume "k1 ~= k2" */
        Object r2a = sa.put(k2, v2);

        Object r2b = sb.put(k2, v2);
        sb.remove(k1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_put_between_c_112(AssociationList sa, AssociationList sb, Object
        k1, Object k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
        null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(k1);
        /*: assume "~(k1 ~= k2)" */
        Object r2a = sa.put(k2, v2);

        Object r2b = sb.put(k2, v2);
        sb.remove(k1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_put_post_s_113(AssociationList sa, AssociationList sb, Object
        k1, Object k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
        null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(k1);
        Object r2a = sa.put(k2, v2);
        /*: assume "k1 ~= k2" */

        Object r2b = sb.put(k2, v2);
        sb.remove(k1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_put_post_c_113(AssociationList sa, AssociationList sb, Object
        k1, Object k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
        null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(k1);
        Object r2a = sa.put(k2, v2);
        /*: assume "~(k1 ~= k2)" */

        Object r2b = sb.put(k2, v2);
```

```
3609        sb.remove(k1);
3610
3611        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3612    }
3613
3614    static void remove_put_pre_s_114(AssociationList sa, AssociationList sb, Object k1,
        Object k2, Object v2)
3615    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
        null &
3616                sa..contents = sb..contents & sa..size = sb..size"
3617        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3618        ensures "True" */
3619    {
3620        /*: assume "k1 ~= k2" */
3621        sa.remove(k1);
3622        sa.put(k2, v2);
3623
3624        sb.put(k2, v2);
3625        sb.remove(k1);
3626
3627        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
3628    }
3629
3630    static void remove_put_pre_c_114(AssociationList sa, AssociationList sb, Object k1,
        Object k2, Object v2)
3631    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
        null &
3632                sa..contents = sb..contents & sa..size = sb..size"
3633        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3634        ensures "True" */
3635    {
3636        /*: assume "~(k1 ~= k2)" */
3637        sa.remove(k1);
3638        sa.put(k2, v2);
3639
3640        sb.put(k2, v2);
3641        sb.remove(k1);
3642
3643        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
3644    }
3645
3646    static void remove_put_between_s_115(AssociationList sa, AssociationList sb, Object
        k1, Object k2, Object v2)
3647    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
        null &
3648                sa..contents = sb..contents & sa..size = sb..size"
3649        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3650        ensures "True" */
3651    {
3652        sa.remove(k1);
3653        /*: assume "k1 ~= k2" */
3654        sa.put(k2, v2);
3655
3656        sb.put(k2, v2);
3657        sb.remove(k1);
3658
3659        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
3660    }
3661
3662    static void remove_put_between_c_115(AssociationList sa, AssociationList sb, Object
        k1, Object k2, Object v2)
3663    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
        null &
3664                sa..contents = sb..contents & sa..size = sb..size"
3665        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
```

```
3666        ensures "True" */
3667    {
3668        sa.remove(k1);
3669        /*: assume "~(k1 ~= k2)" */
3670        sa.put(k2, v2);
3671
3672        sb.put(k2, v2);
3673        sb.remove(k1);
3674
3675        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
3676    }
3677
3678    static void remove_put_post_s_116(AssociationList sa, AssociationList sb, Object
        k1, Object k2, Object v2)
3679    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
        null &
3680                sa..contents = sb..contents & sa..size = sb..size"
3681        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3682        ensures "True" */
3683    {
3684        sa.remove(k1);
3685        sa.put(k2, v2);
3686        /*: assume "k1 ~= k2" */
3687
3688        sb.put(k2, v2);
3689        sb.remove(k1);
3690
3691        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
3692    }
3693
3694    static void remove_put_post_c_116(AssociationList sa, AssociationList sb, Object
        k1, Object k2, Object v2)
3695    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
        null &
3696                sa..contents = sb..contents & sa..size = sb..size"
3697        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3698        ensures "True" */
3699    {
3700        sa.remove(k1);
3701        sa.put(k2, v2);
3702        /*: assume "~(k1 ~= k2)" */
3703
3704        sb.put(k2, v2);
3705        sb.remove(k1);
3706
3707        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
3708    }
3709
3710    static void remove_remove_pre_s_117(AssociationList sa, AssociationList sb, Object
        k1, Object k2)
3711    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3712                sa..contents = sb..contents & sa..size = sb..size"
3713        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3714        ensures "True" */
3715    {
3716        /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
3717        sa.remove(k1);
3718        Object r2a = sa.remove(k2);
3719
3720        Object r2b = sb.remove(k2);
3721        sb.remove(k1);
3722
3723        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3724    }
3725
```

```
    static void remove_remove_pre_c_117(AssociationList sa, AssociationList sb, Object
        k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
        sa.remove(k1);
        Object r2a = sa.remove(k2);

        Object r2b = sb.remove(k2);
        sb.remove(k1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_remove_between_s_118(AssociationList sa, AssociationList sb,
        Object k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(k1);
        /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents))" */
        Object r2a = sa.remove(k2);

        Object r2b = sb.remove(k2);
        sb.remove(k1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_remove_between_c_118(AssociationList sa, AssociationList sb,
        Object k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(k1);
        /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents)))" */
        Object r2a = sa.remove(k2);

        Object r2b = sb.remove(k2);
        sb.remove(k1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_remove_post_s_119(AssociationList sa, AssociationList sb, Object
        k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(k1);
        Object r2a = sa.remove(k2);
        /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents))" */

        Object r2b = sb.remove(k2);
        sb.remove(k1);

```

```
3787        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3788    }
3789
3790    static void remove_remove_post_c_119(AssociationList sa, AssociationList sb, Object
        k1, Object k2)
3791    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3792                  sa..contents = sb..contents & sa..size = sb..size"
3793        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3794        ensures "True" */
3795    {
3796        sa.remove(k1);
3797        Object r2a = sa.remove(k2);
3798        /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents)))" */
3799
3800        Object r2b = sb.remove(k2);
3801        sb.remove(k1);
3802
3803        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3804    }
3805
3806    static void remove_remove_pre_s_120(AssociationList sa, AssociationList sb, Object
        k1, Object k2)
3807    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3808                  sa..contents = sb..contents & sa..size = sb..size"
3809        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3810        ensures "True" */
3811    {
3812        /*: assume "True" */
3813        sa.remove(k1);
3814        sa.remove(k2);
3815
3816        sb.remove(k2);
3817        sb.remove(k1);
3818
3819        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
3820    }
3821
3822    static void remove_remove_pre_c_120(AssociationList sa, AssociationList sb, Object
        k1, Object k2)
3823    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3824                  sa..contents = sb..contents & sa..size = sb..size"
3825        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3826        ensures "True" */
3827    {
3828        /*: assume "~(True)" */
3829        sa.remove(k1);
3830        sa.remove(k2);
3831
3832        sb.remove(k2);
3833        sb.remove(k1);
3834
3835        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
3836    }
3837
3838    static void remove_remove_between_s_121(AssociationList sa, AssociationList sb,
        Object k1, Object k2)
3839    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3840                  sa..contents = sb..contents & sa..size = sb..size"
3841        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3842        ensures "True" */
3843    {
3844        sa.remove(k1);
3845        /*: assume "True" */
3846        sa.remove(k2);
3847
```

```
        sb.remove(k2);
        sb.remove(k1);

        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_remove_between_c_121(AssociationList sa, AssociationList sb,
        Object k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(k1);
        /*: assume "~(True)" */
        sa.remove(k2);

        sb.remove(k2);
        sb.remove(k1);

        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_remove_post_s_122(AssociationList sa, AssociationList sb, Object
        k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(k1);
        sa.remove(k2);
        /*: assume "True" */

        sb.remove(k2);
        sb.remove(k1);

        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_remove_post_c_122(AssociationList sa, AssociationList sb, Object
        k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(k1);
        sa.remove(k2);
        /*: assume "~(True)" */

        sb.remove(k2);
        sb.remove(k1);

        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_size_pre_s_123(AssociationList sa, AssociationList sb, Object k1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(EX v. (k1, v) : sa..contents)" */
        sa.remove(k1);
```

```
3910        int r2a = sa.size();
3911
3912        int r2b = sb.size();
3913        sb.remove(k1);
3914
3915        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3916    }
3917
3918    static void remove_size_pre_c_123(AssociationList sa, AssociationList sb, Object k1)
3919    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
3920                sa..contents = sb..contents & sa..size = sb..size"
3921        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3922        ensures "True" */
3923    {
3924        /*: assume "~(~(EX v. (k1, v) : sa..contents))" */
3925        sa.remove(k1);
3926        int r2a = sa.size();
3927
3928        int r2b = sb.size();
3929        sb.remove(k1);
3930
3931        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3932    }
3933
3934    static void remove_size_between_s_124(AssociationList sa, AssociationList sb,
        Object k1)
3935    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
3936                sa..contents = sb..contents & sa..size = sb..size"
3937        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3938        ensures "True" */
3939    {
3940        sa.remove(k1);
3941        /*: assume "~(EX v. (k1, v) : sa..(old contents))" */
3942        int r2a = sa.size();
3943
3944        int r2b = sb.size();
3945        sb.remove(k1);
3946
3947        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3948    }
3949
3950    static void remove_size_between_c_124(AssociationList sa, AssociationList sb,
        Object k1)
3951    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
3952                sa..contents = sb..contents & sa..size = sb..size"
3953        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3954        ensures "True" */
3955    {
3956        sa.remove(k1);
3957        /*: assume "~(~(EX v. (k1, v) : sa..(old contents)))" */
3958        int r2a = sa.size();
3959
3960        int r2b = sb.size();
3961        sb.remove(k1);
3962
3963        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3964    }
3965
3966    static void remove_size_post_s_125(AssociationList sa, AssociationList sb, Object
        k1)
3967    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
3968                sa..contents = sb..contents & sa..size = sb..size"
3969        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3970        ensures "True" */
3971    {
```

```
        sa.remove(k1);
        int r2a = sa.size();
        /*: assume "~(EX v. (k1, v) : sa..(old contents))" */

        int r2b = sb.size();
        sb.remove(k1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_size_post_c_125(AssociationList sa, AssociationList sb, Object
        k1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(k1);
        int r2a = sa.size();
        /*: assume "~(~(EX v. (k1, v) : sa..(old contents)))" */

        int r2b = sb.size();
        sb.remove(k1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void size_containsKey_pre_s_126(AssociationList sa, AssociationList sb,
        Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        /*: assume "True" */
        int r1a = sa.size();
        boolean r2a = sa.containsKey(k2);

        boolean r2b = sb.containsKey(k2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void size_containsKey_pre_c_126(AssociationList sa, AssociationList sb,
        Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        /*: assume "~(True)" */
        int r1a = sa.size();
        boolean r2a = sa.containsKey(k2);

        boolean r2b = sb.containsKey(k2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void size_containsKey_between_s_127(AssociationList sa, AssociationList sb,
        Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
                  sa..contents = sb..contents & sa..size = sb..size"
```

```
4031        ensures "True" */
4032    {
4033        int r1a = sa.size();
4034        /*: assume "True" */
4035        boolean r2a = sa.containsKey(k2);
4036
4037        boolean r2b = sb.containsKey(k2);
4038        int r1b = sb.size();
4039
4040        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
4041    }
4042
4043    static void size_containsKey_between_c_127(AssociationList sa, AssociationList sb,
            Object k2)
4044    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
4045                  sa..contents = sb..contents & sa..size = sb..size"
4046        ensures "True" */
4047    {
4048        int r1a = sa.size();
4049        /*: assume "~(True)" */
4050        boolean r2a = sa.containsKey(k2);
4051
4052        boolean r2b = sb.containsKey(k2);
4053        int r1b = sb.size();
4054
4055        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
4056    }
4057
4058    static void size_containsKey_post_s_128(AssociationList sa, AssociationList sb,
            Object k2)
4059    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
4060                  sa..contents = sb..contents & sa..size = sb..size"
4061        ensures "True" */
4062    {
4063        int r1a = sa.size();
4064        boolean r2a = sa.containsKey(k2);
4065        /*: assume "True" */
4066
4067        boolean r2b = sb.containsKey(k2);
4068        int r1b = sb.size();
4069
4070        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
4071    }
4072
4073    static void size_containsKey_post_c_128(AssociationList sa, AssociationList sb,
            Object k2)
4074    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
4075                  sa..contents = sb..contents & sa..size = sb..size"
4076        ensures "True" */
4077    {
4078        int r1a = sa.size();
4079        boolean r2a = sa.containsKey(k2);
4080        /*: assume "~(True)" */
4081
4082        boolean r2b = sb.containsKey(k2);
4083        int r1b = sb.size();
4084
4085        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
4086    }
4087
4088    static void size_get_pre_s_129(AssociationList sa, AssociationList sb, Object k2)
```

```
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        /*: assume "True" */
        int r1a = sa.size();
        Object r2a = sa.get(k2);

        Object r2b = sb.get(k2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void size_get_pre_c_129(AssociationList sa, AssociationList sb, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        /*: assume "~(True)" */
        int r1a = sa.size();
        Object r2a = sa.get(k2);

        Object r2b = sb.get(k2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void size_get_between_s_130(AssociationList sa, AssociationList sb, Object
        k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "True" */
        Object r2a = sa.get(k2);

        Object r2b = sb.get(k2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void size_get_between_c_130(AssociationList sa, AssociationList sb, Object
        k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "~(True)" */
        Object r2a = sa.get(k2);

        Object r2b = sb.get(k2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }
```

```
4148    static void size_get_post_s_131(AssociationList sa, AssociationList sb, Object k2)
4149    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
4150                sa..contents = sb..contents & sa..size = sb..size"
4151        ensures "True" */
4152    {
4153        int r1a = sa.size();
4154        Object r2a = sa.get(k2);
4155        /*: assume "True" */
4156
4157        Object r2b = sb.get(k2);
4158        int r1b = sb.size();
4159
4160        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
4161    }
4162
4163    static void size_get_post_c_131(AssociationList sa, AssociationList sb, Object k2)
4164    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
4165                sa..contents = sb..contents & sa..size = sb..size"
4166        ensures "True" */
4167    {
4168        int r1a = sa.size();
4169        Object r2a = sa.get(k2);
4170        /*: assume "~(True)" */
4171
4172        Object r2b = sb.get(k2);
4173        int r1b = sb.size();
4174
4175        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
4176    }
4177
4178    static void size_put_pre_s_132(AssociationList sa, AssociationList sb, Object k2,
            Object v2)
4179    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
4180                sa..contents = sb..contents & sa..size = sb..size"
4181        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4182        ensures "True" */
4183    {
4184        /*: assume "EX v. (k2, v) : sa..contents" */
4185        int r1a = sa.size();
4186        Object r2a = sa.put(k2, v2);
4187
4188        Object r2b = sb.put(k2, v2);
4189        int r1b = sb.size();
4190
4191        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
4192    }
4193
4194    static void size_put_pre_c_132(AssociationList sa, AssociationList sb, Object k2,
            Object v2)
4195    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
4196                sa..contents = sb..contents & sa..size = sb..size"
4197        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4198        ensures "True" */
4199    {
4200        /*: assume "~(EX v. (k2, v) : sa..contents)" */
4201        int r1a = sa.size();
4202        Object r2a = sa.put(k2, v2);
4203
4204        Object r2b = sb.put(k2, v2);
4205        int r1b = sb.size();
4206
```

```
4207        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
4208    }
4209
4210    static void size_put_between_s_133(AssociationList sa, AssociationList sb, Object
            k2, Object v2)
4211    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
4212                  sa..contents = sb..contents & sa..size = sb..size"
4213        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4214        ensures "True" */
4215    {
4216        int r1a = sa.size();
4217        /*: assume "EX v. (k2, v) : sa..contents" */
4218        Object r2a = sa.put(k2, v2);
4219
4220        Object r2b = sb.put(k2, v2);
4221        int r1b = sb.size();
4222
4223        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
4224    }
4225
4226    static void size_put_between_c_133(AssociationList sa, AssociationList sb, Object
            k2, Object v2)
4227    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
4228                  sa..contents = sb..contents & sa..size = sb..size"
4229        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4230        ensures "True" */
4231    {
4232        int r1a = sa.size();
4233        /*: assume "~(EX v. (k2, v) : sa..contents)" */
4234        Object r2a = sa.put(k2, v2);
4235
4236        Object r2b = sb.put(k2, v2);
4237        int r1b = sb.size();
4238
4239        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
4240    }
4241
4242    static void size_put_post_s_134(AssociationList sa, AssociationList sb, Object k2,
            Object v2)
4243    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
4244                  sa..contents = sb..contents & sa..size = sb..size"
4245        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4246        ensures "True" */
4247    {
4248        int r1a = sa.size();
4249        Object r2a = sa.put(k2, v2);
4250        /*: assume "r2a ~= null" */
4251
4252        Object r2b = sb.put(k2, v2);
4253        int r1b = sb.size();
4254
4255        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
4256    }
4257
4258    static void size_put_post_c_134(AssociationList sa, AssociationList sb, Object k2,
            Object v2)
4259    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
4260                  sa..contents = sb..contents & sa..size = sb..size"
4261        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4262        ensures "True" */
4263    {
```

```
        int r1a = sa.size();
        Object r2a = sa.put(k2, v2);
        /*: assume "~(r2a ~= null)" */

        Object r2b = sb.put(k2, v2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void size_put_pre_s_135(AssociationList sa, AssociationList sb, Object k2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "EX v. (k2, v) : sa..contents" */
        int r1a = sa.size();
        sa.put(k2, v2);

        sb.put(k2, v2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void size_put_pre_c_135(AssociationList sa, AssociationList sb, Object k2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(EX v. (k2, v) : sa..contents)" */
        int r1a = sa.size();
        sa.put(k2, v2);

        sb.put(k2, v2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void size_put_between_s_136(AssociationList sa, AssociationList sb, Object
        k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "EX v. (k2, v) : sa..contents" */
        sa.put(k2, v2);

        sb.put(k2, v2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void size_put_between_c_136(AssociationList sa, AssociationList sb, Object
        k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
```

```
4324                 sa..contents = sb..contents & sa..size = sb..size"
4325         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4326         ensures "True" */
4327     {
4328         int r1a = sa.size();
4329         /*: assume "~(EX v. (k2, v) : sa..contents)" */
4330         sa.put(k2, v2);
4331 
4332         sb.put(k2, v2);
4333         int r1b = sb.size();
4334 
4335         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
4336     }
4337 
4338     static void size_put_post_s_137(AssociationList sa, AssociationList sb, Object k2,
        Object v2)
4339     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
4340                 sa..contents = sb..contents & sa..size = sb..size"
4341         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4342         ensures "True" */
4343     {
4344         int r1a = sa.size();
4345         /*: ghost specvar sa__contents :: "(obj * obj) set" = "sa..contents" */
4346         sa.put(k2, v2);
4347         /*: assume "EX v. (k2, v) : sa__contents" */
4348 
4349         sb.put(k2, v2);
4350         int r1b = sb.size();
4351 
4352         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
4353     }
4354 
4355     static void size_put_post_c_137(AssociationList sa, AssociationList sb, Object k2,
        Object v2)
4356     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
4357                 sa..contents = sb..contents & sa..size = sb..size"
4358         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4359         ensures "True" */
4360     {
4361         int r1a = sa.size();
4362         /*: ghost specvar sa__contents :: "(obj * obj) set" = "sa..contents" */
4363         sa.put(k2, v2);
4364         /*: assume "~(EX v. (k2, v) : sa__contents)" */
4365 
4366         sb.put(k2, v2);
4367         int r1b = sb.size();
4368 
4369         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
4370     }
4371 
4372     static void size_remove_pre_s_138(AssociationList sa, AssociationList sb, Object k2)
4373     /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
4374                 sa..contents = sb..contents & sa..size = sb..size"
4375         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4376         ensures "True" */
4377     {
4378         /*: assume "~(EX v. (k2, v) : sa..contents)" */
4379         int r1a = sa.size();
4380         Object r2a = sa.remove(k2);
4381 
4382         Object r2b = sb.remove(k2);
4383         int r1b = sb.size();
4384 
4385         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
```

```
4386    }
4387
4388    static void size_remove_pre_c_138(AssociationList sa, AssociationList sb, Object k2)
4389    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
4390                 sa..contents = sb..contents & sa..size = sb..size"
4391        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4392        ensures "True" */
4393    {
4394        /*: assume "~(~(EX v. (k2, v) : sa..contents))" */
4395        int r1a = sa.size();
4396        Object r2a = sa.remove(k2);
4397
4398        Object r2b = sb.remove(k2);
4399        int r1b = sb.size();
4400
4401        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
4402    }
4403
4404    static void size_remove_between_s_139(AssociationList sa, AssociationList sb,
            Object k2)
4405    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
4406                 sa..contents = sb..contents & sa..size = sb..size"
4407        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4408        ensures "True" */
4409    {
4410        int r1a = sa.size();
4411        /*: assume "~(EX v. (k2, v) : sa..contents)" */
4412        Object r2a = sa.remove(k2);
4413
4414        Object r2b = sb.remove(k2);
4415        int r1b = sb.size();
4416
4417        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
4418    }
4419
4420    static void size_remove_between_c_139(AssociationList sa, AssociationList sb,
            Object k2)
4421    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
4422                 sa..contents = sb..contents & sa..size = sb..size"
4423        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4424        ensures "True" */
4425    {
4426        int r1a = sa.size();
4427        /*: assume "~(~(EX v. (k2, v) : sa..contents))" */
4428        Object r2a = sa.remove(k2);
4429
4430        Object r2b = sb.remove(k2);
4431        int r1b = sb.size();
4432
4433        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
4434    }
4435
4436    static void size_remove_post_s_140(AssociationList sa, AssociationList sb, Object
            k2)
4437    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
4438                 sa..contents = sb..contents & sa..size = sb..size"
4439        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4440        ensures "True" */
4441    {
4442        int r1a = sa.size();
4443        Object r2a = sa.remove(k2);
4444        /*: assume "r2a = null" */
```

```
4445
4446        Object r2b = sb.remove(k2);
4447        int r1b = sb.size();
4448
4449        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
4450    }
4451
4452    static void size_remove_post_c_140(AssociationList sa, AssociationList sb, Object
            k2)
4453    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
4454                  sa..contents = sb..contents & sa..size = sb..size"
4455        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4456        ensures "True" */
4457    {
4458        int r1a = sa.size();
4459        Object r2a = sa.remove(k2);
4460        /*: assume "~(r2a = null)" */
4461
4462        Object r2b = sb.remove(k2);
4463        int r1b = sb.size();
4464
4465        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
4466    }
4467
4468    static void size_remove_pre_s_141(AssociationList sa, AssociationList sb, Object k2)
4469    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
4470                  sa..contents = sb..contents & sa..size = sb..size"
4471        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4472        ensures "True" */
4473    {
4474        /*: assume "~(EX v. (k2, v) : sa..contents)" */
4475        int r1a = sa.size();
4476        sa.remove(k2);
4477
4478        sb.remove(k2);
4479        int r1b = sb.size();
4480
4481        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
4482    }
4483
4484    static void size_remove_pre_c_141(AssociationList sa, AssociationList sb, Object k2)
4485    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
4486                  sa..contents = sb..contents & sa..size = sb..size"
4487        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4488        ensures "True" */
4489    {
4490        /*: assume "~(~(EX v. (k2, v) : sa..contents))" */
4491        int r1a = sa.size();
4492        sa.remove(k2);
4493
4494        sb.remove(k2);
4495        int r1b = sb.size();
4496
4497        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
4498    }
4499
4500    static void size_remove_between_s_142(AssociationList sa, AssociationList sb,
            Object k2)
4501    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
4502                  sa..contents = sb..contents & sa..size = sb..size"
4503        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4504        ensures "True" */
4505    {
```

```
4506        int r1a = sa.size();
4507        /*: assume "~(EX v. (k2, v) : sa..contents)" */
4508        sa.remove(k2);
4509
4510        sb.remove(k2);
4511        int r1b = sb.size();
4512
4513        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
4514    }
4515
4516    static void size_remove_between_c_142(AssociationList sa, AssociationList sb,
        Object k2)
4517    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
4518                sa..contents = sb..contents & sa..size = sb..size"
4519        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4520        ensures "True" */
4521    {
4522        int r1a = sa.size();
4523        /*: assume "~(~(EX v. (k2, v) : sa..contents))" */
4524        sa.remove(k2);
4525
4526        sb.remove(k2);
4527        int r1b = sb.size();
4528
4529        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
4530    }
4531
4532    static void size_remove_post_s_143(AssociationList sa, AssociationList sb, Object
        k2)
4533    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
4534                sa..contents = sb..contents & sa..size = sb..size"
4535        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4536        ensures "True" */
4537    {
4538        int r1a = sa.size();
4539        /*: ghost specvar sa__contents :: "(obj * obj) set" = "sa..contents" */
4540        sa.remove(k2);
4541        /*: assume "~(EX v. (k2, v) : sa__contents)" */
4542
4543        sb.remove(k2);
4544        int r1b = sb.size();
4545
4546        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
4547    }
4548
4549    static void size_remove_post_c_143(AssociationList sa, AssociationList sb, Object
        k2)
4550    /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
4551                sa..contents = sb..contents & sa..size = sb..size"
4552        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4553        ensures "True" */
4554    {
4555        int r1a = sa.size();
4556        /*: ghost specvar sa__contents :: "(obj * obj) set" = "sa..contents" */
4557        sa.remove(k2);
4558        /*: assume "~(~(EX v. (k2, v) : sa__contents))" */
4559
4560        sb.remove(k2);
4561        int r1b = sb.size();
4562
4563        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
4564    }
4565
4566    static void size_size_pre_s_144(AssociationList sa, AssociationList sb)
4567    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
```

```
4568                 sa..contents = sb..contents & sa..size = sb..size"
4569         ensures "True" */
4570     {
4571         /*: assume "True" */
4572         int r1a = sa.size();
4573         int r2a = sa.size();
4574 
4575         int r2b = sb.size();
4576         int r1b = sb.size();
4577 
4578         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
4579     }
4580 
4581     static void size_size_pre_c_144(AssociationList sa, AssociationList sb)
4582     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
4583                 sa..contents = sb..contents & sa..size = sb..size"
4584         ensures "True" */
4585     {
4586         /*: assume "~(True)" */
4587         int r1a = sa.size();
4588         int r2a = sa.size();
4589 
4590         int r2b = sb.size();
4591         int r1b = sb.size();
4592 
4593         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
4594     }
4595 
4596     static void size_size_between_s_145(AssociationList sa, AssociationList sb)
4597     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
4598                 sa..contents = sb..contents & sa..size = sb..size"
4599         ensures "True" */
4600     {
4601         int r1a = sa.size();
4602         /*: assume "True" */
4603         int r2a = sa.size();
4604 
4605         int r2b = sb.size();
4606         int r1b = sb.size();
4607 
4608         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
4609     }
4610 
4611     static void size_size_between_c_145(AssociationList sa, AssociationList sb)
4612     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
4613                 sa..contents = sb..contents & sa..size = sb..size"
4614         ensures "True" */
4615     {
4616         int r1a = sa.size();
4617         /*: assume "~(True)" */
4618         int r2a = sa.size();
4619 
4620         int r2b = sb.size();
4621         int r1b = sb.size();
4622 
4623         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
4624     }
4625 
4626     static void size_size_post_s_146(AssociationList sa, AssociationList sb)
4627     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
4628                 sa..contents = sb..contents & sa..size = sb..size"
```

```
        ensures "True" */
    {
        int r1a = sa.size();
        int r2a = sa.size();
        /*: assume "True" */

        int r2b = sb.size();
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void size_size_post_c_146(AssociationList sa, AssociationList sb)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
        int r2a = sa.size();
        /*: assume "~(True)" */

        int r2b = sb.size();
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

}
```

B.4.3 Inverse Testing Methods

Listing 12. AssociationListInv.java

```
class AssociationListInv {
    static void put_0(AssociationList s, Object k, Object v)
    /*: requires "s ~= null & k ~= null & v ~= null"
        modifies "s..contents", "s..size"
        ensures "True" */
    {
        Object r = s.put(k, v);
        if (r != null) { s.put(k, r); } else { s.remove(k); }

        /*: assert "s..contents = s..(old contents) & s..size = s..(old size)" */
    }

    static void remove_1(AssociationList s, Object k)
    /*: requires "s ~= null & k ~= null"
        modifies "s..contents", "s..size"
        ensures "True" */
    {
        Object r = s.remove(k);
        if (r != null) { s.put(k, r); }

        /*: assert "s..contents = s..(old contents) & s..size = s..(old size)" */
    }

}
```

B.5 HashTable

B.5.1 Data Structure

Listing 13. HashTable.java

```
public /*: claimedby HashTable */ class Node {
    public Object key;
    public Object value;
    public Node next;

    /*: public ghost specvar conts :: "(obj * obj) set" = "{}"
        invariant ConAlloc: "ALL x y. (x,y) : conts --> x : alloc & y : alloc"
        invariant ConNull: "null..conts = {}"
        invariant ConDef: "this ~= null --> conts = {(key, value)} Un next..conts &
            (ALL v. (key, v) ~: next..conts)"
        invariant ConNonNull: "ALL x y. (x, y) : conts --> x ~= null & y ~= null" */
}

public class HashTable {
    /*: public ghost specvar init :: "bool" = "False" */

    private Node[] table = null;

    /*: static specvar abs :: "(int => int)"
        vardefs "abs == (%i1. if (i1 < 0) then (-i1) else i1)"

        static specvar h :: "(obj => int => int)"
        vardefs "h == (%o1. %i1. (abs (hashFunc o1)) mod i1)"
        invariant HashInv: "init --> (ALL k. 0 <= (h k (table..length)) & (h k
            (table..length)) < table..length)"

        invariant TableNotNull: "init --> table ~= null"
        invariant TableHidden: "init --> table : hidden"
        invariant NodeHidden1: "init --> (ALL i. 0 <= i & i < table..length & table.[i]
            ~= null --> table.[i] : hidden)"
        invariant NodeHidden2: "ALL n. n : Node & n : alloc & n ~= null & n..next ~=
            null --> n..next : hidden"
        invariant FirstInjInv: "init --> (ALL i x y. y = x..next & y ~= null & 0 <= i &
            i < table..length --> y ~= table.[i])"
        invariant NextInjInv: "ALL x1 x2 y. y ~= null & y = x1..next & y = x2..next -->
            x1 = x2"
        invariant ElementInjInv: "init --> (ALL ht i j. ht : HashTable & ht : alloc &
            ht..init & 0 <= i & i < ht..table..length & 0 <= j & j < table..length &
            ht..table.[i] = table.[j] & table.[j] ~= null --> ht = this & i = j)"
        invariant Coherence: "init --> (ALL i k v. 0 <= i & i < table..length --> (k,v)
            : table.[i]..conts --> h k (table..length) = i)"
        invariant TableInjInv: "ALL ht. ht..table = table & table ~= null --> ht = this"

        public ghost specvar contents :: "(obj * obj) set" = "{}"
        invariant ContentsDefInv: "init --> contents = {(k,v). (k,v) : table.[(h k
            (table..length))]..conts}"
        public invariant MapInv: "ALL k v0 v1. (k,v0) : contents & (k,v1) : contents
            --> v0 = v1" */

    private int _size;

    /*: public specvar size :: "int"
        vardefs "size == _size"
        invariant CardInv: "_size = card (contents)" */

    public HashTable()
    /*: modifies "contents", "size", "init"
        ensures "init & contents = {} & size = 0" */
    {
        table = new /*: hidden */ Node[11];
        /*: "contents" := "{}" */

        _size = 0;

```

```
54          /*: "init" := "True" */
55
56          /*: note NewNotHT: "table ~: HashTable" */
57
58          {
59              /*: localize */
60              /*: note ElemInj1: "ALL ht1 i j. ht1 : HashTable & ht1 : alloc & ht1..init
                    & 0 <= i & i < ht1..table..length & 0 <= j & j < table..length &
                    ht1..table.[i] = table.[j] & ht1..table.[i] ~= null --> ht1 = this & i =
                    j" */
61              /*: note ElemInj2: "ALL ht2 i j. ht2 : HashTable & ht2 : alloc & ht2..init
                    & 0 <= i & i < table..length & 0 <= j & j < ht2..table..length &
                    table.[i] = ht2..table.[j] & table.[i] ~= null --> this = ht2 & i = j" */
62              /*: note ElemInjOther: "ALL ht1 ht2 i j. ht1 ~= this & ht2 ~= this & ht1 :
                    HashTable & ht1 : alloc & ht1..init & ht2 : HashTable & ht2 : alloc &
                    ht2..init & 0 <= i & i < ht1..table..length & 0 <= j & j <
                    ht2..table..length & ht1..table.[i] = ht2..table.[j] & ht1..table.[i] ~=
                    null --> ht1 = ht2 & i = j" from ElementInjInv, NewNotHT */
63              /*: note ElemInjAll: "theinv ElementInjInv" from ElemInj1, ElemInj2,
                    ElemInjOther */
64          }
65          {
66              /*: localize */
67              /*: note CohThis: "ALL i k v. 0 <= i & i < table..length & (k,v) :
                    table.[i]..conts --> h k (table..length) = i" */
68              /*: note CohOther: "ALL ht. ht ~= this & ht : alloc & ht : HashTable &
                    ht..init --> (ALL i k v. 0 <= i & i < ht..table..length & (k,v) :
                    ht..table.[i]..conts --> h k (ht..table..length) = i)" from Coherence,
                    NewNotHT */
69              /*: note CohAll: "theinv Coherence" from CohThis, CohOther */
70          }
71          {
72              /*: localize */
73              /*: note FirstInjThis: "ALL i x y. y = x..next & y ~= null & 0 <= i & i <
                    table..length --> y ~= table.[i]" */
74              /*: note FirstInjOther: "ALL ht. ht : alloc & ht : HashTable & ht ~= this &
                    ht..init --> (ALL i x y. y = x..next & y ~= null & 0 <= i & i <
                    ht..table..length --> y ~= ht..table.[i])" from FirstInjInv,
                    TableInjInv, NewNotHT */
75              /*: note FirstInjAll: "theinv FirstInjInv" from FirstInjThis, FirstInjOther
                    */
76          }
77          {
78              /*: localize */
79              /*: note TableEmpty: "ALL i. table.[i]..conts = {}" */
80              /*: note ContentsThis: "contents = {(k,v). (k,v) : table.[(h k
                    (table..length))]..conts}" from TableEmpty */
81              /*: note ContentsOther: "ALL ht. ht : alloc & ht : HashTable & ht ~= this
                    --> ht..contents = old (ht..contents)" */
82              /*: note ContentsPost: "theinv ContentsDefInv" from ContentsThis,
                    ContentsOther, ContentsDef, NewNotHT */
83          }
84          {
85              /*: localize */
86              /*: note NodeHiddenThis: "ALL i. 0 <= i & i < table..length & table.[i] ~=
                    null --> table.[i] : hidden" */
87              /*: note NodeHiddenOther: "ALL ht. ht : alloc & ht : HashTable & ht..init &
                    ht ~= this --> (ALL i. 0 <= i & i < ht..table..length & ht..table.[i] ~=
                    null --> ht..table.[i] : hidden)" from NodeHidden1, TableInjInv,
                    NewNotHT */
88              /*: note NodeHiddenAll: "theinv NodeHidden1" from NodeHiddenThis,
                    NodeHiddenOther */
89          }
90          {
91              /*: localize */
```

```
92              /*: note ArrayLength: "0 < table..length" */
93              /*: note HashFuncRel: "h = (%o1. (%i1. (abs (hashFunc o1)) mod i1))"  */
94              /*: note HashThis: "ALL k. 0 <= (h k (table..length)) & (h k
                    (table..length)) < table..length" from ArrayLength, HashFuncRel */
95              /*: note HashOther: "ALL ht. ht ~= this & ht : alloc & ht : HashTable &
                    ht..init --> (ALL k. 0 <= (h k (ht..table..length)) & (h k
                    (ht..table..length)) < ht..table..length)" from HashInv, NewNotHT,
                    TableInjInv */
96              /*: note ShowHashInv: "theinv HashInv" from HashThis, HashOther */
97          }
98          {
99              /*: pickAny x :: obj */
100             {
101                 /*: assuming CardHyp: "x : alloc & x : HashTable" */
102                 {
103                     /*: assuming XIsThisHyp: "x = this" */
104                     /*: note LengthZero: "x.._size = 0" */
105                     /*: note ContentsEmpty: "x..contents = {}" */
106                     /*: note XIsThisCard: "x.._size = card (x..contents)" from
                            XIsThisHyp, LengthZero, ContentsEmpty */
107                 }
108                 {
109                     /*: assuming XNotThisHyp: "x ~= this" */
110                     /*: note XInOldAlloc: "x : old alloc" */
111                     /*: note OldCard: "x..(old HashTable._size) = card (x..(old
                            HashTable.contents))" from CardHyp, XNotThisHyp, XInOldAlloc,
                            CardInv */
112                     /*: note XLengthEq: "x.._size = x..(old HashTable._size)" */
113                     /*: note XContentsUnchanged: "x..contents = x..(old
                            HashTable.contents)" */
114                     /*: note XNotThisCard: "x.._size = card (x..contents)" from
                            XLengthEq, OldCard, XContentsUnchanged */
115                 }
116                 /*: note CardConc: "x.._size = card (x..contents)" from XIsThisCard,
                        XNotThisCard */
117             }
118             /*: note CardPostCond: "x : alloc & x : HashTable --> x.._size = card
                    (x..contents)" forSuch x */
119         }
120     }
121
122     private int compute_hash(Object o1)
123     /*: requires "o1 ~= null & init & theinvs"
124         ensures "result = h o1 (table..length) & 0 <= result & result < table..length &
                alloc = old alloc & theinvs" */
125     {
126         int hc = o1.hashCode();
127
128         if (hc < 0) { hc = -hc; }
129
130         /*: note LengthPos: "0 < table..length" */
131         /*: note ResLt: "(hc mod table..length) < table..length" from TrueBranch,
                FalseBranch, LengthPos */
132
133         return (hc % table.length);
134     }
135
136     public boolean containsKey(Object k0)
137     /*: requires "init & k0 ~= null"
138         ensures "result = (EX v. ((k0, v) : contents))" */
139     {
140         return _containsKey(k0);
141     }
142
143     private boolean _containsKey(Object k0)
```

```
144     /*: requires "k0 ~= null & init & theinvs"
145         ensures "result = (EX v. ((k0, v) : contents)) & theinvs" */
146     {
147         /*: instantiate ContentsThis: "theinv ContentsDefInv" with "this" */
148         /*: mp ContentsRhs: "this : alloc & this : HashTable & init --> contents =
                {(k,v). (k, v) : table.[(h k (table..length))]..conts}" */
149         int hc = compute_hash(k0);
150         boolean res = bucketContainsKey(hc, k0);
151         /*: note HC: "hc = h k0 (table..length)" */
152         /*: note InCon: "res = (EX v. ((k0, v) : table.[hc]..conts))" */
153         /*: note ShowResult: "res = (EX v. ((k0, v) : contents))" from InCon, HC,
                ContentsRhs; */
154         return res;
155     }
156
157     private boolean bucketContainsKey(int bucket_id, Object k0)
158     /*: requires "init & 0 <= bucket_id & bucket_id < table..length & theinvs "
159         ensures "result = (EX v. ((k0, v) : table.[bucket_id]..conts)) & theinvs" */
160     {
161         Node curr = table[bucket_id];
162         while /*: invariant "(EX v. (k0, v) : table.[bucket_id]..conts) = (EX v. (k0,
                v) : curr..conts)" */ (curr != null) {
163             if (curr.key == k0)
164                 return true;
165             curr = curr.next;
166         }
167         return false;
168     }
169
170     public Object remove(Object k0)
171     /*: requires "init & k0 ~= null"
172         modifies "contents", "size"
173         ensures "((EX v. (k0, v) : old contents) --> (k0, result) : old contents &
                contents = old contents - {(k0, result)} & size = old size - 1 & result ~=
                null) & (~(EX v. (k0, v) : old contents) --> contents = old contents & size
                = old size & result = null)" */
174     {
175         if (!_containsKey(k0))
176             return null;
177         else
178             return _remove(k0);
179     }
180
181     private Object removeFirst(Object k0, int hc)
182     /*: requires "init & k0 ~= null & (EX v. (k0, v) : contents) & comment ''KFound''
            (k0 = table.[hc]..key) & comment ''HCProps'' (0 <= hc & hc < table..length & hc
            = h k0 (table..length)) & theinvs"
183         modifies "contents", "size", "conts", "next", "arrayState", "_size"
184         ensures "(k0, result) : old contents & contents = old contents - {(k0, result)}
                & size = old size - 1 & (ALL a i. a ~= table --> a.[i] = old (a.[i])) &
                theinvs" */
185     {
186         /*: ghost specvar v0 :: obj */
187         /*: havoc v0 suchThat InContents: "(k0, v0) : contents" */
188
189         Node f = table[hc];
190         Node second = f.next;
191         f.next = null;
192         /*: "f..conts" := "{(f..key, f..value)}" */
193
194         table[hc] = second;
195         /*: "contents" := "old contents - {(k0, v0)}" */
196
197         _size = _size - 1;
198
```

```
199         /*: note ThisProps: "this : alloc & this : HashTable & init" */
200         /*: note OldContents: "old contents = {(k,v). (k,v) : old (table.[(h k
                (table..length))]..conts)}" from ContentsDefInv, ThisProps */
201         /*: note FNonNull: "f ~= null" */
202         /*: note FProps: "f : Node & f : alloc" from unalloc_lonely, array_pointsto,
                ThisProps */
203         /*: note VFound: "v0 = f..value" from InContents, OldContents, ConDef, KFound,
                FProps, FNonNull, HCProps */
204
205         /*: note Acyclic: "fieldRead (old next) f ~= f" from FNonNull, HCProps,
                FirstInjInv, ThisProps */
206
207         {
208             /*: pickAny ht :: obj suchThat ContentsDefHyp: "ht : alloc & ht : HashTable
                    & ht..init" */
209             /*: note ContentsThis: "ht = this --> ht..contents = {(k,v). (k,v) :
                    ht..table.[(h k (ht..table..length))]..conts}" from OldContents,
                    ElementInjInv, Acyclic, ThisProps, KFound, VFound, ConDef, FProps,
                    FNonNull, HashInv, HCProps */
210             {
211                 /*: assuming NotThisHyp: "ht ~= this" */
212                 /*: note OldHTContents: "fieldRead (old HashTable.contents) ht = {(k,
                        v). (k, v) : (fieldRead (old conts) (arrayRead (old arrayState)
                        (ht..table) (h k (ht..table..length))))}" from ContentsDefHyp,
                        NotThisHyp, ContentsDefInv */
213                 /*: note TableNotEq: "ht..table ~= table" */
214                 /*: note ContentsOther: "ht..contents = {(k, v).  (k, v) :
                        ht..table.[(h k (ht..table..length))]..conts}" from ContentsDefHyp,
                        NotThisHyp, HashInv, ElementInjInv, HCProps, ThisProps, FNonNull,
                        OldHTContents, TableNotEq */
215             }
216             /*: cases "ht = this", "ht ~= this" for ContentsCases: "ht..contents = {(k,
                    v). (k, v) : ht..table.[(h k (ht..table..length))]..conts}" from
                    ContentsThis, ContentsOther */
217             /*: note ContentsDefPostCond: "ht..contents = {(k, v). (k, v) :
                    ht..table.[(h k (ht..table..length))]..conts}" from ContentCases forSuch
                    ht */
218         }
219
220         /*: note CoherencePostCond: "theinv Coherence" from Coherence, Acyclic, ConDef,
                ConNull, FNonNull, TableInjInv, FProps, HCProps */
221
222         /*: note ContentsPost: "contents = old contents - {(f..key, f..value)}" */
223         {
224             /*: pickAny x :: obj */
225             {
226                 /*: assuming CardHyp: "x : alloc & x : HashTable" */
227                 {
228                     /*: note ThisProps: "this : old alloc & this : HashTable" */
229                     /*: note OldCard: "old _size = card (old contents)" from ThisProps,
                            CardInv */
230                     /*: note NewLength: "_size = old _size - 1" */
231                     /*: note NewNotInOld: "(f..key, f..value) : old contents" */
232                     /*: note XIsThisCard: "_size = card (contents)" from OldCard,
                            NewLength, NewNotInOld, ContentsPost */
233                 }
234                 {
235                     /*: assuming XNotThisHyp: "x ~= this" */
236                     /*: note XInOldAlloc: "x : old alloc" */
237                     /*: note OldCard: "x..(old HashTable._size) = card (x..(old
                            HashTable.contents))" from CardHyp, XNotThisHyp, XInOldAlloc,
                            CardInv */
238                     /*: note XLengthEq: "x.._size = x..(old HashTable._size)" */
239                     {
240                         /*: localize */
```

```
241                         /*: note XContentsForw: "ALL y z. (y, z) : x..contents --> (y,
                                z) : x..(old HashTable.contents)" */
242                         /*: note XContentsBack: "ALL y z. (y, z) : x..(old
                                HashTable.contents) --> (y, z) : x..contents" */
243                         /*: note XContentsUnchanged: "x..contents = x..(old
                                HashTable.contents)" from XContentsForw, XContentsBack */
244                     }
245                     /*: note XNotThisCard: "x.._size = card (x..contents)" from
                            XLengthEq, OldCard, XContentsUnchanged */
246                 }
247                 /*: note CardConc: "x.._size = card (x..contents)" from XIsThisCard,
                        XNotThisCard */
248             }
249             /*: note CardPostCond: "x : alloc & x : HashTable --> x.._size = card
                    (x..contents)" forSuch x */
250         }
251
252         return f.value;
253     }
254
255     private Object removeFromBucket(Object k0, int hc)
256     /*: requires "comment ''Init'' init & k0 ~= null & (EX v.(k0, v) : contents) &
            comment ''KNotFound'' (k0 ~= table.[hc]..key) & comment ''HCProps'' (0 <= hc &
            hc < table..length & hc = h k0 (table..length)) & theinvs"
257         modifies "contents", "size", "conts", "next", "arrayState", "_size"
258         ensures "(k0, result) : old contents & contents = old contents - {(k0, result)}
                & size = old size - 1 & (ALL a i. a ~= table --> a.[i] = old (a.[i])) &
                theinvs" */
259     {
260         /*: ghost specvar v0 :: obj */
261         /*: havoc v0 suchThat InContents: "(k0, v0) : contents" */
262
263         Node f = table[hc];
264         Node prev = f;
265
266         /*: note InBucket: "(k0, v0) : prev..conts" from InContents, ContentsDefInv,
                thisNotNull, thisType, Init, HCProps */
267         /*: note PrevNotNull: "prev ~= null" from InBucket, ConDef, ConNull */
268
269         /*: "prev..conts" := "prev..conts - {(k0, v0)}" */
270         /*: "contents"   := "old contents - {(k0, v0)}" */
271
272         Node curr = prev.next;
273
274         /*: note PrevHidden: "prev : hidden" from NodeHidden1, thisNotNull, thisType,
                PrevNotNull, Init, HCProps */
275         /*: note ConPreLoop: "ALL n. n : Node & n : alloc & n ~= null & n ~= prev -->
                n..conts = {(n..key, n..value)} Un n..next..conts & (ALL v.(n..key, v) ~:
                n..next..conts)" from ConDef, FirstInjInv, Init, HCProps, thisNotNull,
                thisType, PrevNotNull */
276
277         /*: note ConUnchangedPreLoop: "ALL ht i. ht ~= this & ht : HashTable & ht :
                alloc & ht..init & 0 <= i & i < ht..table..length --> ht..table.[i]..conts =
                old (ht..table.[i]..conts)" from ElementInjInv, thisType, PrevNotNull, Init,
                HCProps */
278
279         while /*: invariant "prev : Node & prev : alloc & prev ~= null & prev : hidden
                & comment ''PrevCon'' (prev..conts = fieldRead (old conts) prev - {(k0,
                v0)}) & comment ''PrevNot'' (ALL v.(prev..key, v) ~: prev..next..conts) &
                comment ''CurrProps'' (curr : Node & curr : alloc) & comment ''CurrNotNull''
                (curr ~= null) & comment ''PrevCurr'' (prev..next = curr & prev ~= curr) &
                contents = old contents - {(k0, v0)} & (k0, v0) : curr..conts & comment
                ''ConDefInv'' (ALL n. n : Node & n : alloc & n ~= null & n ~= prev -->
                n..conts = {(n..key, n..value)} Un n..next..conts & (ALL v.(n..key, v) ~:
                n..next..conts)) & comment ''ConLoop'' (ALL n. n..conts = old (n..conts) |
```

```
    n..conts = old (n..conts) - {(k0, v0)}) & (null..conts = {}) & comment
    ''FConInv'' (f..conts = (fieldRead (old conts) f) - {(k0, v0)}) & comment
    ''ConUnchanged'' (ALL ht i. ht ~= this & ht : HashTable & ht : alloc &
    ht..init & 0 <= i & i < ht..table..length --> ht..table.[i]..conts = old
    (ht..table.[i]..conts))" */ (curr.key != k0) {
280     /*: "curr..conts" := "curr..conts - {(k0, v0)}" */
281
282     /*: note CurrCon: "curr..conts = fieldRead (old conts) curr - {(k0, v0)}" */
283     /*: note PrevIsNot: "prev..key ~= k0" */
284     /*: note OldConDef: "fieldRead (old conts) prev = {(prev..key,
            prev..value)} Un fieldRead (old conts) (prev..next)" */
285     /*: note PrevConDef: "prev..conts = {(prev..key, prev..value)} Un
            prev..next..conts" from PrevCurr, PrevCon, CurrCon, OldConDef, PrevIsNot
            */
286
287     prev = curr;
288     curr = curr.next;
289
290     /*: note FConLem: "f..conts = (fieldRead (old conts) f) - {(k0, v0)}" from
            FConInv */
291
292     /*: note ConExceptPrev: "ALL n. n : Node & n : alloc & n ~= null & n ~=
            prev --> n..conts = {(n..key, n..value)} Un n..next..conts & (ALL
            v.(n..key, v) ~: n..next..conts)" from PrevConDef, PrevNot, ConDefInv,
            PrevCurr, NextInjInv, CurrNotNull */
293 }
294
295 Node tmp = curr.next;
296 prev.next = tmp;
297 curr.next = null;
298
299 /*: "curr..conts" := "{(curr..key, curr..value)}" */
300
301 _size = _size - 1;
302
303 {
304     /*: pickAny x :: obj suchThat xHyp: "x : Node & x : alloc & x ~= null" */
305     {
306         /*: assuming xIsPrev: "x = prev" */
307         /*: note nextNotCurr: "fieldRead (old next) curr ~= curr" from
                NextInjInv, CurrNotNull, PrevCurr, CurrProps */
308         /*: note prevNextCon: "prev..next..conts = fieldRead (old conts)
                (prev..next)" */
309         /*: note prevOldCon: "fieldRead (old conts) prev = {(prev..key,
                prev..value)} Un fieldRead (old conts) curr" */
310         /*: note currOldCon: "fieldRead (old conts) curr = {(curr..key,
                curr..value)} Un fieldRead (old conts) (fieldRead (old next) curr)"
                */
311         /*: note prevKeyNotK0: "prev..key ~= k0" */
312         {
313             /*: pickAny k :: obj, v :: obj suchThat ForwHyp: "(k, v) :
                    x..conts" */
314             /*: note kNotK0: "k ~= k0" */
315             /*: note currKeyIsK0: "curr..key = k0" */
316             /*: note ForwCase: "(k, v) : {(x..key, x..value)} Un
                    x..next..conts" from xHyp, xIsPrev, ForwHyp, PrevCurr,
                    nextNotCurr, PrevCon, prevNextCon, prevOldCon, currOldCon,
                    prevKeyNotK0, kNotK0, currKeyIsK0 forSuch k, v */
317         }
318         /*: note BackCase: "ALL k v. (k, v) : {(x..key, x..value)} Un
                x..next..conts --> (k, v) : x..conts" */
319         /*: note xCon: "x..conts = {(x..key, x..value)} Un x..next..conts" from
                ForwCase, BackCase */
320     }
```

```
321             /*: cases "x = curr", "x = prev", "x ~= curr & x ~= prev" for XCon:
                    "x..conts = {(x..key, x..value)} Un x..next..conts" */
322             /*: note ConPost: "x..conts = {(x..key, x..value)} Un x..next..conts & (ALL
                    v. (x..key, v) ~: x..next..conts)" forSuch x */
323         }
324
325         {
326             /*: pickAny ht :: obj suchThat CohHyp: "ht : alloc & ht : HashTable &
                    ht..init" */
327             {
328                 /*: pickAny i :: int, k :: obj, v :: obj suchThat InnerHyp: "0 <= i & i
                        < ht..table..length & (k, v) : ht..table.[i]..conts" */
329                 /*: note NotCurr: "ht..table.[i] ~= curr" */
330                 /*: note InnerConc: "h k (ht..table..length) = i" from CohHyp,
                        InnerHyp, Coherence, NotCurr, ConLoop forSuch i, k, v */
331             }
332             /*: note CoherencePost: "(ALL i k v. 0 <= i & i < ht..table..length --> (k,
                    v) : ht..table.[i]..conts --> h k (ht..table..length) = i)" from
                    InnerConc forSuch ht */
333         }
334
335         {
336             /*: pickAny x :: obj suchThat ContentsDefHyp: "x : alloc & x : HashTable &
                    x..init" */
337             /*: note OldXContents: "fieldRead (old HashTable.contents) x = {(k, v). (k,
                    v) : fieldRead (old conts) (arrayRead (old arrayState) (x..table) (h k
                    (x..table..length)))}" from ContentsDefHyp, ContentsDefInv */
338             {
339                 /*: assuming XNotThisHyp: "x ~= this" */
340                 /*: note NotCurr: "ALL i. 0 <= i & i < x..table..length -->
                        x..table.[i] ~= curr" */
341                 /*: note ConXUnchanged: "ALL i.0 <= i & i < x..table..length -->
                        x..table.[i]..conts = fieldRead (old conts) (arrayRead (old
                        arrayState) (x..table) i)" from ContentsDefHyp, XNotThisHyp,
                        NotCurr, ConUnchanged */
342                 /*: note LengthLemma: "ALL k.0 <= (h k (x..table..length)) & (h k
                        (x..table..length)) < (x..table..length)" from ContentsDefHyp,
                        HashInv */
343                 /*: note XNotThisCase: "x..contents = {(k, v). (k, v) : x..table.[(h k
                        (x..table..length))]..conts}" from XNotThisHyp, OldXContents,
                        LengthLemma, ConXUnchanged */
344             }
345             {
346                 /*: assuming XIsThisHyp: "x = this" */
347                 /*: note OldContents: "old contents = {(k,v). (k,v) : old (table.[(h k
                        (table..length))]..conts)}" */
348                 {
349                     /*: pickAny k :: obj, v :: obj suchThat ForwHyp: "(k, v) :
                            contents" */
350                     /*: note NotCurr: "table.[(h k (table..length))] ~= curr" from
                            FirstInjInv, ContentsDefHyp,XIsThisHyp, PrevCurr, CurrNotNull,
                            HashInv */
351                     /*: note ForwCase: "(k, v) : table.[(h k (table..length))]..conts"
                            from ForwHyp, OldContents, NotCurr, ConLoop forSuch k, v */
352                 }
353                 {
354                     /*: pickAny k :: obj, v :: obj suchThat BackHyp: "(k, v) :
                            table.[(h k (table..length))]..conts" */
355                     /*: note NotCurr: "table.[(h k (table..length))] ~= curr" from
                            FirstInjInv, ContentsDefHyp, XIsThisHyp, PrevCurr, CurrNotNull,
                            HashInv */
356                     /*: note BackCase: "(k, v) : contents" from BackHyp, OldContents,
                            NotCurr, ConLoop, FConInv, HCProps forSuch k, v */
357                 }
```

```
358                 /*: note XIsThisCase: "contents = {(k, v). (k, v) : table.[(h k
                        (table..length))]..conts}" from ForwCase, BackCase */
359             }
360             /*: note ContentsDefPost: "x..contents = {(k, v). (k, v) : x..table.[(h k
                    (x..table..length))]..conts}" from XNotThisCase, XIsThisCase forSuch x */
361         }
362
363         /*: note ContentsPost: "contents = old contents - {(curr..key, curr..value)}" */
364         {
365             /*: pickAny x :: obj */
366             {
367                 /*: assuming CardHyp: "x : alloc & x : HashTable" */
368                 {
369                     /*: note ThisProps: "this : old alloc & this : HashTable" */
370                     /*: note OldCard: "old _size = card (old contents)" from ThisProps,
                            CardInv */
371                     /*: note NewLength: "_size = old _size - 1" */
372                     /*: note NewNotInOld: "(curr..key, curr..value) : old contents" */
373                     /*: note XIsThisCard: "_size = card (contents)" from OldCard,
                            NewLength, NewNotInOld, ContentsPost */
374                 }
375                 {
376                     /*: assuming XNotThisHyp: "x ~= this" */
377                     /*: note XInOldAlloc: "x : old alloc" */
378                     /*: note OldCard: "x..(old HashTable._size) = card (x..(old
                            HashTable.contents))" from CardHyp, XNotThisHyp, XInOldAlloc,
                            CardInv */
379                     /*: note XLengthEq: "x.._size = x..(old HashTable._size)" */
380                     {
381                         /*: localize */
382                         /*: note XContentsForw: "ALL y z. (y, z) : x..contents --> (y,
                                z) : x..(old HashTable.contents)" */
383                         /*: note XContentsBack: "ALL y z. (y, z) : x..(old
                                HashTable.contents) --> (y, z) : x..contents" */
384                         /*: note XContentsUnchanged: "x..contents = x..(old
                                HashTable.contents)" from XContentsForw, XContentsBack */
385                     }
386                     /*: note XNotThisCard: "x.._size = card (x..contents)" from
                            XLengthEq, OldCard, XContentsUnchanged */
387                 }
388                 /*: note CardConc: "x.._size = card (x..contents)" from XIsThisCard,
                        XNotThisCard */
389             }
390             /*: note CardPostCond: "x : alloc & x : HashTable --> x.._size = card
                    (x..contents)" forSuch x */
391         }
392
393         return curr.value;
394     }
395
396     private Object _remove(Object k0)
397     /*: requires "(comment ''Init'' init) & k0 ~= null & (EX v. (k0, v) : contents) &
            theinvs"
398         modifies "contents", "size", "conts", "next", "arrayState", "_size"
399         ensures "(k0, result) : old contents & contents = old contents - {(k0, result)}
                & size = old size - 1 & (ALL a i. a ~= table --> a.[i] = old (a.[i])) &
                theinvs" */
400     {
401         /*: ghost specvar v0 :: obj */
402         /*: havoc v0 suchThat KeyInContents: "(k0, v0) : contents" */
403
404         int hc = compute_hash(k0);
405         Node f = table[hc];
406
407         /*: note ThisProps: "this : alloc & this : HashTable" */
```

```
        /*: note HCDef: "hc = h k0 (table..length)" */
        /*: note KeyInBucket: "(k0, v0) : table.[hc]..conts" from HCDef, KeyInContents,
            ContentsDef, Init, ThisProps */

        if (f.key == k0) {
            return removeFirst(k0, hc);
        } else {
            return removeFromBucket(k0, hc);
        }
    }

    private void _add(Object k0, Object v0)
    /*: requires "comment ''Init'' init & k0 ~= null & v0 ~= null & ~(EX v. (k0, v) :
        contents) & theinvs"
        modifies "contents", "size", "arrayState", "new..conts", "new..next",
            "new..value", "new..key", "_size"
        ensures  "contents = old contents Un {(k0, v0)} & size = old size + 1 & (ALL a
            i. a ~= table --> a.[i] = old (a.[i])) & theinvs" */
    {
        int hc = compute_hash(k0);
        Node n = new /*: hidden */ Node();
        n.key = k0;
        n.value = v0;
        Node first = table[hc];
        n.next = first;
        /*: "n..conts" := "{(k0, v0)} Un first..conts" */
        table[hc] = n;
        /*: "contents" := "(old contents) Un {(k0, v0)}" */

        _size = _size + 1;

        /*: note NewNotHT: "n ~: HashTable" */
        /*: note ThisProps: "this : alloc & this : old alloc & this : HashTable" */
        /*: note HCBounds: "0 <= hc & hc < table..length" */
        /*: note NewOldNEq: "n ~= first" */

        /*: note ThisTableNotNull: "table ~= null" */

        /*: note OldNotRefInTable: "ALL ht i. ht : alloc & ht : HashTable & ht..init &
            0 <= i & i < ht..table..length & first ~= null --> ht..table.[i] ~= first"
            from OldNotRefInTable, Init, ThisProps, HCBounds, ElementInjInvHash,
            NewOldNEq, TableInjInvHash, NewNotHT, ThisTableNotNull */

        /*: note HashPost: "theinv HashInv" from HashPost, HashInv, NewNotHT */

        /*: note KeyAlloc: "k0 : alloc" */
        /*: note ValAlloc: "v0 : alloc" */
        /*: note AllocChanged: "alloc = old alloc Un {n}" */
        /*: note FirstProps: "first : alloc & first : Node & first ~= n" from
            unalloc_lonely, AllocChanged, ThisProps, array_pointsto, NewNotHT */

        /*: note ConAllocLemma: "theinv ConAlloc" from ConAllocLemma, ConAlloc,
            KeyAlloc, ValAlloc, FirstProps */

        /*: note NewHidden2: "n..next ~= null --> n..next : hidden" from NewHidden2,
            NodeHidden1, ThisProps, Init, HCBounds */
        /*: note OldHidden2: "ALL m. m ~= n & m : Node & m : alloc & m ~= null &
            m..next ~= null --> m..next : hidden" */
        /*: note AllHidden2: "theinv NodeHidden2" from AllHidden2, NewHidden2,
            OldHidden2 */

        /*: note NewHidden: "n : hidden" */
        /*: note ThisHidden1: "ALL i. 0 <= i & i < table..length & table.[i] ~= null
            --> table.[i] : hidden" from ThisHidden1, NodeHidden1, NewHidden, ThisProps,
            Init */
```

```
459         /*: note OtherHidden1: "ALL ht. ht ~= this & ht : alloc & ht : HashTable &
                ht..init --> (ALL i. 0 <= i & i < ht..table..length & ht..table.[i] ~= null
                --> ht..table.[i] : hidden)" from OtherHidden1, NodeHidden1, NewNotHT */
460         /*: note Hidden1All: "theinv NodeHidden1" from Hidden1All, ThisHidden1,
                OtherHidden1 */
461
462         /*: note AllocChange: "Object.alloc = old Object.alloc Un {n}" */
463         /*: note HProp: "hc = h k0 (table..Array.length)" */
464
465         /*: note NewNotRefThisArr: "ALL i. 0 <= i & i < table..length --> (arrayRead
                (old arrayState) table i) ~= n" */
466         /*: note NewNotRefArray: "ALL a i. 0 <= i & i < a..length --> (arrayRead (old
                arrayState) a i) ~= n" */
467
468         /*: note NewNotAlloc: "n ~: old alloc" */
469         /*: note NewNotRefByNext: "ALL x. x..next ~= n" from NewOldNEq, NewNotAlloc,
                unalloc_lonely, NewNotRefByNext */
470
471         /*: note NotInOldContents: "ALL v. (k0, v) ~: old contents" */
472         /*: note NotInOldConFirst: "ALL v. (k0, v) ~: (fieldRead (old conts) first)"
                from NotInOldConFirst, NotInOldContents, ContentsDef, ThisProps, Init, HProp
                */
473         /*: note FirstNotN: "first ~= n" */
474         /*: note NewConDef: "theinv ConDef" from NewConDef, ConDef, NewNotRefByNext,
                FirstNotN, NotInOldConFirst */
475
476         /*: note ElemInj: "theinv ElementInjInv" from ElementInjInv, ThisProps,
                NewNotHT, NewNotRefArray, TableInjInv, ThisTableNotNull */
477
478         {
479             /*: pickAny ht :: obj */
480             {
481                 /*: assuming h1: "ht : alloc & ht : HashTable & ht..init" */
482                 {
483                     /*: pickAny i :: int, k :: obj, v :: obj */
484                     /*: note g1: "arrayRead (old arrayState) (ht..table) i ~= n" */
485                     /*: note g2: "ht : old alloc" */
486                     {
487                         /*: assuming h2: "0 <= i & i < ht..table..length & (k,v) :
                                ht..table.[i]..conts" */
488                         {
489                             /*: assuming h3: "ht = this" */
490                             /*: note c5: "h k (ht..table..length) = i" from c3, h1, h2,
                                    h3, Coherence, g1, g2, HProp */
491                         }
492                         {
493                             /*: assuming h4: "ht ~= this" */
494                             /*: note g3: "ht..table ~= table" */
495                             /*: note c4: "h k (ht..table..length) = i" from c4, h1, h2,
                                    h4, Coherence, g1, g2, g3 */
496                         }
497                         /*: note c3: "h k (ht..table..length) = i" from c4, c5 */
498                     }
499                     /*: note c2: "0 <= i & i < ht..table..length --> (k,v) :
                            ht..table.[i]..conts --> h k (ht..table..length) = i" forSuch i,
                            k, v */
500                 }
501                 /*: note c1: "ALL i k v. 0 <= i & i < ht..table..length --> (k,v) :
                        ht..table.[i]..conts --> h k (ht..table..length) = i" */
502             }
503             /*: note CohPost: "ht : alloc & ht : HashTable & ht..init --> (ALL i k v. 0
                    <= i & i < ht..table..length --> (k,v) : ht..table.[i]..conts --> h k
                    (ht..table..length) = i)" forSuch ht */
504         }
505
```

```
506         {
507             /*: pickAny ht :: obj */
508             {
509                 /*: assuming NewContentsHyp: "ht : alloc & ht : HashTable & ht..init" */
510                 /*: note ThisContents: "contents = {(k,v). (k,v) : table.[(h k
                        (table..length))]..conts}" from ThisContents, HProp,
                        NewNotRefThisArr, HashInv, ContentsDefInv,  ThisProps, Init */
511                 {
512                     /*: assuming OtherHyp: "ht ~= this" */
513                     /*: note OtherContents: "ht..contents = {(k,v). (k,v) :
                            ht..table.[(h k (ht..table..length))]..conts}" from OtherHyp,
                            NewContentsHyp, ContentsDefInvHash, NewNotHT, TableInjInvHash,
                            NewNotRefArray, TableNotNullHash, HashInv */
514                 }
515
516                 /*: cases "ht = this", "ht ~= this" for NewContentsCases: "ht..contents
                        = {(k,v). (k,v) : ht..table.[(h k (ht..table..length))]..conts}"
                        from ThisContents, OtherContents */
517                 /*: note NewContentsConc: "ht..contents = {(k,v). (k,v) : ht..table.[(h
                        k (ht..table..length))]..conts}" */
518             }
519             /*: note NewContentsDef: "ht : alloc & ht : HashTable & ht..init -->
                    ht..contents = {(k,v). (k,v) : ht..table.[(h k
                    (ht..table..length))]..conts}" forSuch ht */
520         }
521
522         /*: note OldNotRefByNext: "ALL x. first ~= null --> old (x..next) ~= first"
                from FirstInjInv, Init, HCBounds, OldNotRefByNext, ThisProps */
523
524         /*: note NewFirstInj: "theinv FirstInjInv" from FirstInjInv, NewNotRefByNext,
                OldNotRefByNext, TableInjInv, OldNotRefInTable, HCBounds, NewFirstInj,
                ThisProps, ElementInjInv, NewNotHT */
525
526         /*: note NewNextInj: "theinv NextInjInv" from NextInjInv, OldNotRefByNext,
                NewNextInj */
527
528         /*: note ContentsPost: "contents = old contents Un {(k0, v0)}" */
529         {
530             /*: pickAny x :: obj */
531             {
532                 /*: assuming CardHyp: "x : alloc & x : HashTable" */
533                 {
534                     /*: note ThisProps: "this : old alloc & this : HashTable" */
535                     /*: note OldCard: "old _size = card (old contents)" from ThisProps,
                            CardInv */
536                     /*: note NewLength: "_size = old _size + 1" */
537                     /*: note NewNotInOld: "(k0, v0) ~: old contents" */
538                     /*: note XIsThisCard: "_size = card (contents)" from OldCard,
                            NewLength, NewNotInOld, ContentsPost */
539                 }
540                 {
541                     /*: assuming XNotThisHyp: "x ~= this" */
542                     /*: note XInOldAlloc: "x : old alloc" */
543                     /*: note OldCard: "x..(old HashTable._size) = card (x..(old
                            HashTable.contents))" from CardHyp, XNotThisHyp, XInOldAlloc,
                            CardInv */
544                     /*: note XLengthEq: "x.._size = x..(old HashTable._size)" */
545                     {
546                         /*: localize */
547                         /*: note XContentsForw: "ALL y z. (y, z) : x..contents --> (y,
                                z) : x..(old HashTable.contents)" */
548                         /*: note XContentsBack: "ALL y z. (y, z) : x..(old
                                HashTable.contents) --> (y, z) : x..contents" */
549                         /*: note XContentsUnchanged: "x..contents = x..(old
                                HashTable.contents)" from XContentsForw, XContentsBack */
```

```
                    }
                    /*: note XNotThisCard: "x.._size = card (x..contents)" from
                        XLengthEq, OldCard, XContentsUnchanged */
                }
                /*: note CardConc: "x.._size = card (x..contents)" from XIsThisCard,
                    XNotThisCard */
            }
            /*: note CardPostCond: "x : alloc & x : HashTable --> x.._size = card
                (x..contents)" forSuch x */
        }
    }

    public Object put(Object k0, Object v0)
    /*: requires "init & k0 ~= null & v0 ~= null"
        modifies "contents", "size"
        ensures "((EX v. (k0, v) : old contents) --> (k0, result) : old contents &
            contents = old contents - {(k0, result)} Un {(k0, v0)} & size = old size &
            result ~= null) & (~(EX v. (k0, v) : old contents) --> contents = old
            contents Un {(k0, v0)} & size = old size + 1 & result = null) & (result ~=
            null --> (k0, result) : old contents & old contents = contents - {(k0, v0)}
            Un {(k0, result)} & old size = size) & (result = null --> ~(EX v. (k0, v) :
            old contents) & old contents = contents - {(k0, v0)} & old size = size - 1)"
            */
    {
        if (_containsKey(k0)) {
            Object v1 = _remove(k0);
            _add(k0, v0);
            return v1;
        } else {
            _add(k0, v0);
            return null;
        }
    }

    public Object get(Object k0)
    /*: requires "init & k0 ~= null"
        ensures "(result ~= null --> (k0, result) : contents) & (result = null --> ~(EX
            v. (k0, v) : contents))" */
    {
        /*: instantiate "theinv ContentsDefInv" with "this" */
        /*: mp ThisContentsDef: "this : alloc & this : HashTable & init --> contents =
            {(k, v). (k, v) : table.[(h k (table..length))]..conts}" */
        int hc = compute_hash(k0);
        Node curr = table[hc];

        /*: note HCDef: "hc = h k0 (table..length)" */
        /*: note InCurr: "ALL v. (((k0, v) : contents) = ((k0, v) : curr..conts))" from
            ThisContentsDef, HCDef */

        while /*: invariant "ALL v. ((k0, v) : contents) = ((k0, v) : curr..conts)" */
            (curr != null) {
            if (curr.key == k0) {
                return curr.value;
            }

            curr = curr.next;
        }
        return null;
    }

    public int size()
    /*: ensures "result = size" */
    {
        return _size;
    }
```

```
}
```

B.5.2 Commutativity Testing Methods

Listing 14. HashTableComm.java

```
class HashTableComm {
    static void containsKey_containsKey_pre_s_0(HashTable sa, HashTable sb, Object k1,
        Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        /*: assume "True" */
        boolean r1a = sa.containsKey(k1);
        boolean r2a = sa.containsKey(k2);

        boolean r2b = sb.containsKey(k2);
        boolean r1b = sb.containsKey(k1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void containsKey_containsKey_pre_c_0(HashTable sa, HashTable sb, Object k1,
        Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        /*: assume "~(True)" */
        boolean r1a = sa.containsKey(k1);
        boolean r2a = sa.containsKey(k2);

        boolean r2b = sb.containsKey(k2);
        boolean r1b = sb.containsKey(k1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void containsKey_containsKey_between_s_1(HashTable sa, HashTable sb, Object
        k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        boolean r1a = sa.containsKey(k1);
        /*: assume "True" */
        boolean r2a = sa.containsKey(k2);

        boolean r2b = sb.containsKey(k2);
        boolean r1b = sb.containsKey(k1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void containsKey_containsKey_between_c_1(HashTable sa, HashTable sb, Object
        k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
```

```
49                  sa..contents = sb..contents & sa..size = sb..size"
50          ensures "True" */
51      {
52          boolean r1a = sa.containsKey(k1);
53          /*: assume "~(True)" */
54          boolean r2a = sa.containsKey(k2);
55
56          boolean r2b = sb.containsKey(k2);
57          boolean r1b = sb.containsKey(k1);
58
59          /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
60      }
61
62      static void containsKey_containsKey_post_s_2(HashTable sa, HashTable sb, Object k1,
            Object k2)
63      /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
64                  sa..contents = sb..contents & sa..size = sb..size"
65          ensures "True" */
66      {
67          boolean r1a = sa.containsKey(k1);
68          boolean r2a = sa.containsKey(k2);
69          /*: assume "True" */
70
71          boolean r2b = sb.containsKey(k2);
72          boolean r1b = sb.containsKey(k1);
73
74          /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
75      }
76
77      static void containsKey_containsKey_post_c_2(HashTable sa, HashTable sb, Object k1,
            Object k2)
78      /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
79                  sa..contents = sb..contents & sa..size = sb..size"
80          ensures "True" */
81      {
82          boolean r1a = sa.containsKey(k1);
83          boolean r2a = sa.containsKey(k2);
84          /*: assume "~(True)" */
85
86          boolean r2b = sb.containsKey(k2);
87          boolean r1b = sb.containsKey(k1);
88
89          /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
90      }
91
92      static void containsKey_get_pre_s_3(HashTable sa, HashTable sb, Object k1, Object
            k2)
93      /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
94                  sa..contents = sb..contents & sa..size = sb..size"
95          ensures "True" */
96      {
97          /*: assume "True" */
98          boolean r1a = sa.containsKey(k1);
99          Object r2a = sa.get(k2);
100
101         Object r2b = sb.get(k2);
102         boolean r1b = sb.containsKey(k1);
103
```

```
104         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
105     }
106
107     static void containsKey_get_pre_c_3(HashTable sa, HashTable sb, Object k1, Object
            k2)
108     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
109                 sa..contents = sb..contents & sa..size = sb..size"
110         ensures "True" */
111     {
112         /*: assume "~(True)" */
113         boolean r1a = sa.containsKey(k1);
114         Object r2a = sa.get(k2);
115
116         Object r2b = sb.get(k2);
117         boolean r1b = sb.containsKey(k1);
118
119         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
120     }
121
122     static void containsKey_get_between_s_4(HashTable sa, HashTable sb, Object k1,
            Object k2)
123     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
124                 sa..contents = sb..contents & sa..size = sb..size"
125         ensures "True" */
126     {
127         boolean r1a = sa.containsKey(k1);
128         /*: assume "True" */
129         Object r2a = sa.get(k2);
130
131         Object r2b = sb.get(k2);
132         boolean r1b = sb.containsKey(k1);
133
134         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
135     }
136
137     static void containsKey_get_between_c_4(HashTable sa, HashTable sb, Object k1,
            Object k2)
138     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
139                 sa..contents = sb..contents & sa..size = sb..size"
140         ensures "True" */
141     {
142         boolean r1a = sa.containsKey(k1);
143         /*: assume "~(True)" */
144         Object r2a = sa.get(k2);
145
146         Object r2b = sb.get(k2);
147         boolean r1b = sb.containsKey(k1);
148
149         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
150     }
151
152     static void containsKey_get_post_s_5(HashTable sa, HashTable sb, Object k1, Object
            k2)
153     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
154                 sa..contents = sb..contents & sa..size = sb..size"
155         ensures "True" */
156     {
```

```
157         boolean r1a = sa.containsKey(k1);
158         Object r2a = sa.get(k2);
159         /*: assume "True" */
160 
161         Object r2b = sb.get(k2);
162         boolean r1b = sb.containsKey(k1);
163 
164         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
165     }
166 
167     static void containsKey_get_post_c_5(HashTable sa, HashTable sb, Object k1, Object
            k2)
168     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
169                 sa..contents = sb..contents & sa..size = sb..size"
170         ensures "True" */
171     {
172         boolean r1a = sa.containsKey(k1);
173         Object r2a = sa.get(k2);
174         /*: assume "~(True)" */
175 
176         Object r2b = sb.get(k2);
177         boolean r1b = sb.containsKey(k1);
178 
179         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
180     }
181 
182     static void containsKey_put_pre_s_6(HashTable sa, HashTable sb, Object k1, Object
            k2, Object v2)
183     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null & v2 ~= null &
184                 sa..contents = sb..contents & sa..size = sb..size"
185         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
186         ensures "True" */
187     {
188         /*: assume "k1 ~= k2 | (EX v. (k1, v) : sa..contents)" */
189         boolean r1a = sa.containsKey(k1);
190         Object r2a = sa.put(k2, v2);
191 
192         Object r2b = sb.put(k2, v2);
193         boolean r1b = sb.containsKey(k1);
194 
195         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
196     }
197 
198     static void containsKey_put_pre_c_6(HashTable sa, HashTable sb, Object k1, Object
            k2, Object v2)
199     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null & v2 ~= null &
200                 sa..contents = sb..contents & sa..size = sb..size"
201         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
202         ensures "True" */
203     {
204         /*: assume "~(k1 ~= k2 | (EX v. (k1, v) : sa..contents))" */
205         boolean r1a = sa.containsKey(k1);
206         Object r2a = sa.put(k2, v2);
207 
208         Object r2b = sb.put(k2, v2);
209         boolean r1b = sb.containsKey(k1);
210 
211         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
```

```
212     }
213
214     static void containsKey_put_between_s_7(HashTable sa, HashTable sb, Object k1,
            Object k2, Object v2)
215     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null & v2 ~= null &
216                 sa..contents = sb..contents & sa..size = sb..size"
217         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
218         ensures "True" */
219     {
220         boolean r1a = sa.containsKey(k1);
221         /*: assume "k1 ~= k2 | r1a" */
222         Object r2a = sa.put(k2, v2);
223
224         Object r2b = sb.put(k2, v2);
225         boolean r1b = sb.containsKey(k1);
226
227         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
228     }
229
230     static void containsKey_put_between_c_7(HashTable sa, HashTable sb, Object k1,
            Object k2, Object v2)
231     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null & v2 ~= null &
232                 sa..contents = sb..contents & sa..size = sb..size"
233         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
234         ensures "True" */
235     {
236         boolean r1a = sa.containsKey(k1);
237         /*: assume "~(k1 ~= k2 | r1a)" */
238         Object r2a = sa.put(k2, v2);
239
240         Object r2b = sb.put(k2, v2);
241         boolean r1b = sb.containsKey(k1);
242
243         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
244     }
245
246     static void containsKey_put_post_s_8(HashTable sa, HashTable sb, Object k1, Object
            k2, Object v2)
247     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null & v2 ~= null &
248                 sa..contents = sb..contents & sa..size = sb..size"
249         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
250         ensures "True" */
251     {
252         boolean r1a = sa.containsKey(k1);
253         Object r2a = sa.put(k2, v2);
254         /*: assume "k1 ~= k2 | r1a" */
255
256         Object r2b = sb.put(k2, v2);
257         boolean r1b = sb.containsKey(k1);
258
259         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
260     }
261
262     static void containsKey_put_post_c_8(HashTable sa, HashTable sb, Object k1, Object
            k2, Object v2)
263     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null & v2 ~= null &
264                 sa..contents = sb..contents & sa..size = sb..size"
265         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
```

```
266         ensures "True" */
267     {
268         boolean r1a = sa.containsKey(k1);
269         Object r2a = sa.put(k2, v2);
270         /*: assume "~(k1 ~= k2 | r1a)" */
271 
272         Object r2b = sb.put(k2, v2);
273         boolean r1b = sb.containsKey(k1);
274 
275         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
276     }
277 
278     static void containsKey_put_pre_s_9(HashTable sa, HashTable sb, Object k1, Object
            k2, Object v2)
279     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null & v2 ~= null &
280                 sa..contents = sb..contents & sa..size = sb..size"
281         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
282         ensures "True" */
283     {
284         /*: assume "k1 ~= k2 | (EX v. (k1, v) : sa..contents)" */
285         boolean r1a = sa.containsKey(k1);
286         sa.put(k2, v2);
287 
288         sb.put(k2, v2);
289         boolean r1b = sb.containsKey(k1);
290 
291         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
292     }
293 
294     static void containsKey_put_pre_c_9(HashTable sa, HashTable sb, Object k1, Object
            k2, Object v2)
295     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null & v2 ~= null &
296                 sa..contents = sb..contents & sa..size = sb..size"
297         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
298         ensures "True" */
299     {
300         /*: assume "~(k1 ~= k2 | (EX v. (k1, v) : sa..contents))" */
301         boolean r1a = sa.containsKey(k1);
302         sa.put(k2, v2);
303 
304         sb.put(k2, v2);
305         boolean r1b = sb.containsKey(k1);
306 
307         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
308     }
309 
310     static void containsKey_put_between_s_10(HashTable sa, HashTable sb, Object k1,
            Object k2, Object v2)
311     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null & v2 ~= null &
312                 sa..contents = sb..contents & sa..size = sb..size"
313         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
314         ensures "True" */
315     {
316         boolean r1a = sa.containsKey(k1);
317         /*: assume "k1 ~= k2 | r1a" */
318         sa.put(k2, v2);
319 
320         sb.put(k2, v2);
321         boolean r1b = sb.containsKey(k1);
322 
323         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
```

```
324     }
325
326     static void containsKey_put_between_c_10(HashTable sa, HashTable sb, Object k1,
            Object k2, Object v2)
327     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null & v2 ~= null &
328                 sa..contents = sb..contents & sa..size = sb..size"
329         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
330         ensures "True" */
331     {
332         boolean r1a = sa.containsKey(k1);
333         /*: assume "~(k1 ~= k2 | r1a)" */
334         sa.put(k2, v2);
335
336         sb.put(k2, v2);
337         boolean r1b = sb.containsKey(k1);
338
339         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
340     }
341
342     static void containsKey_put_post_s_11(HashTable sa, HashTable sb, Object k1, Object
            k2, Object v2)
343     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null & v2 ~= null &
344                 sa..contents = sb..contents & sa..size = sb..size"
345         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
346         ensures "True" */
347     {
348         boolean r1a = sa.containsKey(k1);
349         sa.put(k2, v2);
350         /*: assume "k1 ~= k2 | r1a" */
351
352         sb.put(k2, v2);
353         boolean r1b = sb.containsKey(k1);
354
355         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
356     }
357
358     static void containsKey_put_post_c_11(HashTable sa, HashTable sb, Object k1, Object
            k2, Object v2)
359     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null & v2 ~= null &
360                 sa..contents = sb..contents & sa..size = sb..size"
361         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
362         ensures "True" */
363     {
364         boolean r1a = sa.containsKey(k1);
365         sa.put(k2, v2);
366         /*: assume "~(k1 ~= k2 | r1a)" */
367
368         sb.put(k2, v2);
369         boolean r1b = sb.containsKey(k1);
370
371         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
372     }
373
374     static void containsKey_remove_pre_s_12(HashTable sa, HashTable sb, Object k1,
            Object k2)
375     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
376                 sa..contents = sb..contents & sa..size = sb..size"
377         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
378         ensures "True" */
379     {
380         /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
```

```
381         boolean r1a = sa.containsKey(k1);
382         Object r2a = sa.remove(k2);
383 
384         Object r2b = sb.remove(k2);
385         boolean r1b = sb.containsKey(k1);
386 
387         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
388     }
389 
390     static void containsKey_remove_pre_c_12(HashTable sa, HashTable sb, Object k1,
            Object k2)
391     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
392                 sa..contents = sb..contents & sa..size = sb..size"
393         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
394         ensures "True" */
395     {
396         /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
397         boolean r1a = sa.containsKey(k1);
398         Object r2a = sa.remove(k2);
399 
400         Object r2b = sb.remove(k2);
401         boolean r1b = sb.containsKey(k1);
402 
403         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
404     }
405 
406     static void containsKey_remove_between_s_13(HashTable sa, HashTable sb, Object k1,
            Object k2)
407     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
408                 sa..contents = sb..contents & sa..size = sb..size"
409         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
410         ensures "True" */
411     {
412         boolean r1a = sa.containsKey(k1);
413         /*: assume "k1 ~= k2 | ~r1a" */
414         Object r2a = sa.remove(k2);
415 
416         Object r2b = sb.remove(k2);
417         boolean r1b = sb.containsKey(k1);
418 
419         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
420     }
421 
422     static void containsKey_remove_between_c_13(HashTable sa, HashTable sb, Object k1,
            Object k2)
423     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
424                 sa..contents = sb..contents & sa..size = sb..size"
425         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
426         ensures "True" */
427     {
428         boolean r1a = sa.containsKey(k1);
429         /*: assume "~(k1 ~= k2 | ~r1a)" */
430         Object r2a = sa.remove(k2);
431 
432         Object r2b = sb.remove(k2);
433         boolean r1b = sb.containsKey(k1);
434 
435         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
```

```
436     }
437
438     static void containsKey_remove_post_s_14(HashTable sa, HashTable sb, Object k1,
            Object k2)
439     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
440                 sa..contents = sb..contents & sa..size = sb..size"
441         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
442         ensures "True" */
443     {
444         boolean r1a = sa.containsKey(k1);
445         Object r2a = sa.remove(k2);
446         /*: assume "k1 ~= k2 | ~r1a" */
447
448         Object r2b = sb.remove(k2);
449         boolean r1b = sb.containsKey(k1);
450
451         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
452     }
453
454     static void containsKey_remove_post_c_14(HashTable sa, HashTable sb, Object k1,
            Object k2)
455     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
456                 sa..contents = sb..contents & sa..size = sb..size"
457         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
458         ensures "True" */
459     {
460         boolean r1a = sa.containsKey(k1);
461         Object r2a = sa.remove(k2);
462         /*: assume "~(k1 ~= k2 | ~r1a)" */
463
464         Object r2b = sb.remove(k2);
465         boolean r1b = sb.containsKey(k1);
466
467         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
468     }
469
470     static void containsKey_remove_pre_s_15(HashTable sa, HashTable sb, Object k1,
            Object k2)
471     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
472                 sa..contents = sb..contents & sa..size = sb..size"
473         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
474         ensures "True" */
475     {
476         /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
477         boolean r1a = sa.containsKey(k1);
478         sa.remove(k2);
479
480         sb.remove(k2);
481         boolean r1b = sb.containsKey(k1);
482
483         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
484     }
485
486     static void containsKey_remove_pre_c_15(HashTable sa, HashTable sb, Object k1,
            Object k2)
487     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
488                 sa..contents = sb..contents & sa..size = sb..size"
489         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
490         ensures "True" */
```

```
491     {
492         /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
493         boolean r1a = sa.containsKey(k1);
494         sa.remove(k2);
495 
496         sb.remove(k2);
497         boolean r1b = sb.containsKey(k1);
498 
499         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
500     }
501 
502     static void containsKey_remove_between_s_16(HashTable sa, HashTable sb, Object k1,
        Object k2)
503     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
504                 sa..contents = sb..contents & sa..size = sb..size"
505         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
506         ensures "True" */
507     {
508         boolean r1a = sa.containsKey(k1);
509         /*: assume "k1 ~= k2 | ~r1a" */
510         sa.remove(k2);
511 
512         sb.remove(k2);
513         boolean r1b = sb.containsKey(k1);
514 
515         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
516     }
517 
518     static void containsKey_remove_between_c_16(HashTable sa, HashTable sb, Object k1,
        Object k2)
519     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
520                 sa..contents = sb..contents & sa..size = sb..size"
521         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
522         ensures "True" */
523     {
524         boolean r1a = sa.containsKey(k1);
525         /*: assume "~(k1 ~= k2 | ~r1a)" */
526         sa.remove(k2);
527 
528         sb.remove(k2);
529         boolean r1b = sb.containsKey(k1);
530 
531         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
532     }
533 
534     static void containsKey_remove_post_s_17(HashTable sa, HashTable sb, Object k1,
        Object k2)
535     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
536                 sa..contents = sb..contents & sa..size = sb..size"
537         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
538         ensures "True" */
539     {
540         boolean r1a = sa.containsKey(k1);
541         sa.remove(k2);
542         /*: assume "k1 ~= k2 | ~r1a" */
543 
544         sb.remove(k2);
545         boolean r1b = sb.containsKey(k1);
546 
547         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
548     }
549 
```

```
    static void containsKey_remove_post_c_17(HashTable sa, HashTable sb, Object k1,
        Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        boolean r1a = sa.containsKey(k1);
        sa.remove(k2);
        /*: assume "~(k1 ~= k2 | ~r1a)" */

        sb.remove(k2);
        boolean r1b = sb.containsKey(k1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void containsKey_size_pre_s_18(HashTable sa, HashTable sb, Object k1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        /*: assume "True" */
        boolean r1a = sa.containsKey(k1);
        int r2a = sa.size();

        int r2b = sb.size();
        boolean r1b = sb.containsKey(k1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void containsKey_size_pre_c_18(HashTable sa, HashTable sb, Object k1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        /*: assume "~(True)" */
        boolean r1a = sa.containsKey(k1);
        int r2a = sa.size();

        int r2b = sb.size();
        boolean r1b = sb.containsKey(k1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void containsKey_size_between_s_19(HashTable sa, HashTable sb, Object k1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        boolean r1a = sa.containsKey(k1);
        /*: assume "True" */
        int r2a = sa.size();

        int r2b = sb.size();
        boolean r1b = sb.containsKey(k1);

```

```
608         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
609     }
610
611     static void containsKey_size_between_c_19(HashTable sa, HashTable sb, Object k1)
612     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            &
613                 sa..contents = sb..contents & sa..size = sb..size"
614         ensures "True" */
615     {
616         boolean r1a = sa.containsKey(k1);
617         /*: assume "~(True)" */
618         int r2a = sa.size();
619
620         int r2b = sb.size();
621         boolean r1b = sb.containsKey(k1);
622
623         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
624     }
625
626     static void containsKey_size_post_s_20(HashTable sa, HashTable sb, Object k1)
627     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            &
628                 sa..contents = sb..contents & sa..size = sb..size"
629         ensures "True" */
630     {
631         boolean r1a = sa.containsKey(k1);
632         int r2a = sa.size();
633         /*: assume "True" */
634
635         int r2b = sb.size();
636         boolean r1b = sb.containsKey(k1);
637
638         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
639     }
640
641     static void containsKey_size_post_c_20(HashTable sa, HashTable sb, Object k1)
642     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            &
643                 sa..contents = sb..contents & sa..size = sb..size"
644         ensures "True" */
645     {
646         boolean r1a = sa.containsKey(k1);
647         int r2a = sa.size();
648         /*: assume "~(True)" */
649
650         int r2b = sb.size();
651         boolean r1b = sb.containsKey(k1);
652
653         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
654     }
655
656     static void get_containsKey_pre_s_21(HashTable sa, HashTable sb, Object k1, Object
            k2)
657     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
658                 sa..contents = sb..contents & sa..size = sb..size"
659         ensures "True" */
660     {
661         /*: assume "True" */
662         Object r1a = sa.get(k1);
663         boolean r2a = sa.containsKey(k2);
```

```
664
665         boolean r2b = sb.containsKey(k2);
666         Object r1b = sb.get(k1);
667
668         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
669     }
670
671     static void get_containsKey_pre_c_21(HashTable sa, HashTable sb, Object k1, Object
            k2)
672     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
673                 sa..contents = sb..contents & sa..size = sb..size"
674         ensures "True" */
675     {
676         /*: assume "~(True)" */
677         Object r1a = sa.get(k1);
678         boolean r2a = sa.containsKey(k2);
679
680         boolean r2b = sb.containsKey(k2);
681         Object r1b = sb.get(k1);
682
683         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
684     }
685
686     static void get_containsKey_between_s_22(HashTable sa, HashTable sb, Object k1,
            Object k2)
687     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
688                 sa..contents = sb..contents & sa..size = sb..size"
689         ensures "True" */
690     {
691         Object r1a = sa.get(k1);
692         /*: assume "True" */
693         boolean r2a = sa.containsKey(k2);
694
695         boolean r2b = sb.containsKey(k2);
696         Object r1b = sb.get(k1);
697
698         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
699     }
700
701     static void get_containsKey_between_c_22(HashTable sa, HashTable sb, Object k1,
            Object k2)
702     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
703                 sa..contents = sb..contents & sa..size = sb..size"
704         ensures "True" */
705     {
706         Object r1a = sa.get(k1);
707         /*: assume "~(True)" */
708         boolean r2a = sa.containsKey(k2);
709
710         boolean r2b = sb.containsKey(k2);
711         Object r1b = sb.get(k1);
712
713         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
714     }
715
716     static void get_containsKey_post_s_23(HashTable sa, HashTable sb, Object k1, Object
            k2)
```

```
717     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
718                 sa..contents = sb..contents & sa..size = sb..size"
719         ensures "True" */
720     {
721         Object r1a = sa.get(k1);
722         boolean r2a = sa.containsKey(k2);
723         /*: assume "True" */
724 
725         boolean r2b = sb.containsKey(k2);
726         Object r1b = sb.get(k1);
727 
728         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
729     }
730 
731     static void get_containsKey_post_c_23(HashTable sa, HashTable sb, Object k1, Object
            k2)
732     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
733                 sa..contents = sb..contents & sa..size = sb..size"
734         ensures "True" */
735     {
736         Object r1a = sa.get(k1);
737         boolean r2a = sa.containsKey(k2);
738         /*: assume "~(True)" */
739 
740         boolean r2b = sb.containsKey(k2);
741         Object r1b = sb.get(k1);
742 
743         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
744     }
745 
746     static void get_get_pre_s_24(HashTable sa, HashTable sb, Object k1, Object k2)
747     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
748                 sa..contents = sb..contents & sa..size = sb..size"
749         ensures "True" */
750     {
751         /*: assume "True" */
752         Object r1a = sa.get(k1);
753         Object r2a = sa.get(k2);
754 
755         Object r2b = sb.get(k2);
756         Object r1b = sb.get(k1);
757 
758         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
759     }
760 
761     static void get_get_pre_c_24(HashTable sa, HashTable sb, Object k1, Object k2)
762     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
763                 sa..contents = sb..contents & sa..size = sb..size"
764         ensures "True" */
765     {
766         /*: assume "~(True)" */
767         Object r1a = sa.get(k1);
768         Object r2a = sa.get(k2);
769 
770         Object r2b = sb.get(k2);
771         Object r1b = sb.get(k1);
772 
```

```
        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void get_get_between_s_25(HashTable sa, HashTable sb, Object k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        Object r1a = sa.get(k1);
        /*: assume "True" */
        Object r2a = sa.get(k2);

        Object r2b = sb.get(k2);
        Object r1b = sb.get(k1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void get_get_between_c_25(HashTable sa, HashTable sb, Object k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        Object r1a = sa.get(k1);
        /*: assume "~(True)" */
        Object r2a = sa.get(k2);

        Object r2b = sb.get(k2);
        Object r1b = sb.get(k1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void get_get_post_s_26(HashTable sa, HashTable sb, Object k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        Object r1a = sa.get(k1);
        Object r2a = sa.get(k2);
        /*: assume "True" */

        Object r2b = sb.get(k2);
        Object r1b = sb.get(k1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void get_get_post_c_26(HashTable sa, HashTable sb, Object k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        Object r1a = sa.get(k1);
        Object r2a = sa.get(k2);
        /*: assume "~(True)" */

```

```
        Object r2b = sb.get(k2);
        Object r1b = sb.get(k1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void get_put_pre_s_27(HashTable sa, HashTable sb, Object k1, Object k2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "k1 ~= k2 | (k1, v2) : sa..contents" */
        Object r1a = sa.get(k1);
        Object r2a = sa.put(k2, v2);

        Object r2b = sb.put(k2, v2);
        Object r1b = sb.get(k1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void get_put_pre_c_27(HashTable sa, HashTable sb, Object k1, Object k2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(k1 ~= k2 | (k1, v2) : sa..contents)" */
        Object r1a = sa.get(k1);
        Object r2a = sa.put(k2, v2);

        Object r2b = sb.put(k2, v2);
        Object r1b = sb.get(k1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void get_put_between_s_28(HashTable sa, HashTable sb, Object k1, Object k2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.get(k1);
        /*: assume "k1 ~= k2 | r1a = v2" */
        Object r2a = sa.put(k2, v2);

        Object r2b = sb.put(k2, v2);
        Object r1b = sb.get(k1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

```

```
884     static void get_put_between_c_28(HashTable sa, HashTable sb, Object k1, Object k2,
            Object v2)
885     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null & v2 ~= null &
886                 sa..contents = sb..contents & sa..size = sb..size"
887         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
888         ensures "True" */
889     {
890         Object r1a = sa.get(k1);
891         /*: assume "~(k1 ~= k2 | r1a = v2)" */
892         Object r2a = sa.put(k2, v2);
893
894         Object r2b = sb.put(k2, v2);
895         Object r1b = sb.get(k1);
896
897         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
898     }
899
900     static void get_put_post_s_29(HashTable sa, HashTable sb, Object k1, Object k2,
            Object v2)
901     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null & v2 ~= null &
902                 sa..contents = sb..contents & sa..size = sb..size"
903         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
904         ensures "True" */
905     {
906         Object r1a = sa.get(k1);
907         Object r2a = sa.put(k2, v2);
908         /*: assume "k1 ~= k2 | r1a = v2" */
909
910         Object r2b = sb.put(k2, v2);
911         Object r1b = sb.get(k1);
912
913         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
914     }
915
916     static void get_put_post_c_29(HashTable sa, HashTable sb, Object k1, Object k2,
            Object v2)
917     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null & v2 ~= null &
918                 sa..contents = sb..contents & sa..size = sb..size"
919         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
920         ensures "True" */
921     {
922         Object r1a = sa.get(k1);
923         Object r2a = sa.put(k2, v2);
924         /*: assume "~(k1 ~= k2 | r1a = v2)" */
925
926         Object r2b = sb.put(k2, v2);
927         Object r1b = sb.get(k1);
928
929         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
930     }
931
932     static void get_put_pre_s_30(HashTable sa, HashTable sb, Object k1, Object k2,
            Object v2)
933     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null & v2 ~= null &
934                 sa..contents = sb..contents & sa..size = sb..size"
935         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
936         ensures "True" */
937     {
```

```
938         /*: assume "k1 ~= k2 | (k1, v2) : sa..contents" */
939         Object r1a = sa.get(k1);
940         sa.put(k2, v2);
941 
942         sb.put(k2, v2);
943         Object r1b = sb.get(k1);
944 
945         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
946     }
947 
948     static void get_put_pre_c_30(HashTable sa, HashTable sb, Object k1, Object k2,
        Object v2)
949     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null & v2 ~= null &
950                 sa..contents = sb..contents & sa..size = sb..size"
951         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
952         ensures "True" */
953     {
954         /*: assume "~(k1 ~= k2 | (k1, v2) : sa..contents)" */
955         Object r1a = sa.get(k1);
956         sa.put(k2, v2);
957 
958         sb.put(k2, v2);
959         Object r1b = sb.get(k1);
960 
961         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
962     }
963 
964     static void get_put_between_s_31(HashTable sa, HashTable sb, Object k1, Object k2,
        Object v2)
965     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null & v2 ~= null &
966                 sa..contents = sb..contents & sa..size = sb..size"
967         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
968         ensures "True" */
969     {
970         Object r1a = sa.get(k1);
971         /*: assume "k1 ~= k2 | r1a = v2" */
972         sa.put(k2, v2);
973 
974         sb.put(k2, v2);
975         Object r1b = sb.get(k1);
976 
977         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
978     }
979 
980     static void get_put_between_c_31(HashTable sa, HashTable sb, Object k1, Object k2,
        Object v2)
981     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null & v2 ~= null &
982                 sa..contents = sb..contents & sa..size = sb..size"
983         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
984         ensures "True" */
985     {
986         Object r1a = sa.get(k1);
987         /*: assume "~(k1 ~= k2 | r1a = v2)" */
988         sa.put(k2, v2);
989 
990         sb.put(k2, v2);
991         Object r1b = sb.get(k1);
992 
993         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
994     }
995 
```

```
    static void get_put_post_s_32(HashTable sa, HashTable sb, Object k1, Object k2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.get(k1);
        sa.put(k2, v2);
        /*: assume "k1 ~= k2 | r1a = v2" */

        sb.put(k2, v2);
        Object r1b = sb.get(k1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void get_put_post_c_32(HashTable sa, HashTable sb, Object k1, Object k2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.get(k1);
        sa.put(k2, v2);
        /*: assume "~(k1 ~= k2 | r1a = v2)" */

        sb.put(k2, v2);
        Object r1b = sb.get(k1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void get_remove_pre_s_33(HashTable sa, HashTable sb, Object k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
        Object r1a = sa.get(k1);
        Object r2a = sa.remove(k2);

        Object r2b = sb.remove(k2);
        Object r1b = sb.get(k1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void get_remove_pre_c_33(HashTable sa, HashTable sb, Object k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
        Object r1a = sa.get(k1);
        Object r2a = sa.remove(k2);

```

```
        Object r2b = sb.remove(k2);
        Object r1b = sb.get(k1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void get_remove_between_s_34(HashTable sa, HashTable sb, Object k1, Object
        k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
                    sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.get(k1);
        /*: assume "k1 ~= k2 | r1a = null" */
        Object r2a = sa.remove(k2);

        Object r2b = sb.remove(k2);
        Object r1b = sb.get(k1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void get_remove_between_c_34(HashTable sa, HashTable sb, Object k1, Object
        k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
                    sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.get(k1);
        /*: assume "~(k1 ~= k2 | r1a = null)" */
        Object r2a = sa.remove(k2);

        Object r2b = sb.remove(k2);
        Object r1b = sb.get(k1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void get_remove_post_s_35(HashTable sa, HashTable sb, Object k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
                    sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.get(k1);
        Object r2a = sa.remove(k2);
        /*: assume "k1 ~= k2 | r1a = null" */

        Object r2b = sb.remove(k2);
        Object r1b = sb.get(k1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void get_remove_post_c_35(HashTable sa, HashTable sb, Object k1, Object k2)
```

```
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.get(k1);
        Object r2a = sa.remove(k2);
        /*: assume "~(k1 ~= k2 | r1a = null)" */

        Object r2b = sb.remove(k2);
        Object r1b = sb.get(k1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void get_remove_pre_s_36(HashTable sa, HashTable sb, Object k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
        Object r1a = sa.get(k1);
        sa.remove(k2);

        sb.remove(k2);
        Object r1b = sb.get(k1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void get_remove_pre_c_36(HashTable sa, HashTable sb, Object k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
        Object r1a = sa.get(k1);
        sa.remove(k2);

        sb.remove(k2);
        Object r1b = sb.get(k1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void get_remove_between_s_37(HashTable sa, HashTable sb, Object k1, Object
        k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.get(k1);
        /*: assume "k1 ~= k2 | r1a = null" */
        sa.remove(k2);

        sb.remove(k2);
        Object r1b = sb.get(k1);
```

```
1168
1169        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1170    }
1171
1172    static void get_remove_between_c_37(HashTable sa, HashTable sb, Object k1, Object
        k2)
1173    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
1174                sa..contents = sb..contents & sa..size = sb..size"
1175     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1176     ensures "True" */
1177    {
1178        Object r1a = sa.get(k1);
1179        /*: assume "~(k1 ~= k2 | r1a = null)" */
1180        sa.remove(k2);
1181
1182        sb.remove(k2);
1183        Object r1b = sb.get(k1);
1184
1185        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1186    }
1187
1188    static void get_remove_post_s_38(HashTable sa, HashTable sb, Object k1, Object k2)
1189    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
1190                sa..contents = sb..contents & sa..size = sb..size"
1191     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1192     ensures "True" */
1193    {
1194        Object r1a = sa.get(k1);
1195        sa.remove(k2);
1196        /*: assume "k1 ~= k2 | r1a = null" */
1197
1198        sb.remove(k2);
1199        Object r1b = sb.get(k1);
1200
1201        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1202    }
1203
1204    static void get_remove_post_c_38(HashTable sa, HashTable sb, Object k1, Object k2)
1205    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
1206                sa..contents = sb..contents & sa..size = sb..size"
1207     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1208     ensures "True" */
1209    {
1210        Object r1a = sa.get(k1);
1211        sa.remove(k2);
1212        /*: assume "~(k1 ~= k2 | r1a = null)" */
1213
1214        sb.remove(k2);
1215        Object r1b = sb.get(k1);
1216
1217        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1218    }
1219
1220    static void get_size_pre_s_39(HashTable sa, HashTable sb, Object k1)
1221    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        &
1222                sa..contents = sb..contents & sa..size = sb..size"
1223     ensures "True" */
1224    {
1225        /*: assume "True" */
1226        Object r1a = sa.get(k1);
1227        int r2a = sa.size();
```

```
1228
1229        int r2b = sb.size();
1230        Object r1b = sb.get(k1);
1231
1232        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
1233    }
1234
1235    static void get_size_pre_c_39(HashTable sa, HashTable sb, Object k1)
1236    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            &
1237                sa..contents = sb..contents & sa..size = sb..size"
1238        ensures "True" */
1239    {
1240        /*: assume "~(True)" */
1241        Object r1a = sa.get(k1);
1242        int r2a = sa.size();
1243
1244        int r2b = sb.size();
1245        Object r1b = sb.get(k1);
1246
1247        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
1248    }
1249
1250    static void get_size_between_s_40(HashTable sa, HashTable sb, Object k1)
1251    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            &
1252                sa..contents = sb..contents & sa..size = sb..size"
1253        ensures "True" */
1254    {
1255        Object r1a = sa.get(k1);
1256        /*: assume "True" */
1257        int r2a = sa.size();
1258
1259        int r2b = sb.size();
1260        Object r1b = sb.get(k1);
1261
1262        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
1263    }
1264
1265    static void get_size_between_c_40(HashTable sa, HashTable sb, Object k1)
1266    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            &
1267                sa..contents = sb..contents & sa..size = sb..size"
1268        ensures "True" */
1269    {
1270        Object r1a = sa.get(k1);
1271        /*: assume "~(True)" */
1272        int r2a = sa.size();
1273
1274        int r2b = sb.size();
1275        Object r1b = sb.get(k1);
1276
1277        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
1278    }
1279
1280    static void get_size_post_s_41(HashTable sa, HashTable sb, Object k1)
1281    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            &
1282                sa..contents = sb..contents & sa..size = sb..size"
1283        ensures "True" */
1284    {
```

```
        Object r1a = sa.get(k1);
        int r2a = sa.size();
        /*: assume "True" */

        int r2b = sb.size();
        Object r1b = sb.get(k1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void get_size_post_c_41(HashTable sa, HashTable sb, Object k1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        &
                sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        Object r1a = sa.get(k1);
        int r2a = sa.size();
        /*: assume "~(True)" */

        int r2b = sb.size();
        Object r1b = sb.get(k1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void put_containsKey_pre_s_42(HashTable sa, HashTable sb, Object k1, Object
        v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "k1 ~= k2 | (EX v. (k1, v) : sa..contents)" */
        Object r1a = sa.put(k1, v1);
        boolean r2a = sa.containsKey(k2);

        boolean r2b = sb.containsKey(k2);
        Object r1b = sb.put(k1, v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void put_containsKey_pre_c_42(HashTable sa, HashTable sb, Object k1, Object
        v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(k1 ~= k2 | (EX v. (k1, v) : sa..contents))" */
        Object r1a = sa.put(k1, v1);
        boolean r2a = sa.containsKey(k2);

        boolean r2b = sb.containsKey(k2);
        Object r1b = sb.put(k1, v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }
```

```
1341
1342    static void put_containsKey_between_s_43(HashTable sa, HashTable sb, Object k1,
            Object v1, Object k2)
1343    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & v1 ~= null & k2 ~= null &
1344                    sa..contents = sb..contents & sa..size = sb..size"
1345        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1346        ensures "True" */
1347    {
1348        Object r1a = sa.put(k1, v1);
1349        /*: assume "k1 ~= k2 | r1a ~= null" */
1350        boolean r2a = sa.containsKey(k2);
1351
1352        boolean r2b = sb.containsKey(k2);
1353        Object r1b = sb.put(k1, v1);
1354
1355        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
1356    }
1357
1358    static void put_containsKey_between_c_43(HashTable sa, HashTable sb, Object k1,
            Object v1, Object k2)
1359    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & v1 ~= null & k2 ~= null &
1360                    sa..contents = sb..contents & sa..size = sb..size"
1361        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1362        ensures "True" */
1363    {
1364        Object r1a = sa.put(k1, v1);
1365        /*: assume "~(k1 ~= k2 | r1a ~= null)" */
1366        boolean r2a = sa.containsKey(k2);
1367
1368        boolean r2b = sb.containsKey(k2);
1369        Object r1b = sb.put(k1, v1);
1370
1371        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
1372    }
1373
1374    static void put_containsKey_post_s_44(HashTable sa, HashTable sb, Object k1, Object
            v1, Object k2)
1375    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & v1 ~= null & k2 ~= null &
1376                    sa..contents = sb..contents & sa..size = sb..size"
1377        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1378        ensures "True" */
1379    {
1380        Object r1a = sa.put(k1, v1);
1381        boolean r2a = sa.containsKey(k2);
1382        /*: assume "k1 ~= k2 | r1a ~= null" */
1383
1384        boolean r2b = sb.containsKey(k2);
1385        Object r1b = sb.put(k1, v1);
1386
1387        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
1388    }
1389
1390    static void put_containsKey_post_c_44(HashTable sa, HashTable sb, Object k1, Object
            v1, Object k2)
1391    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & v1 ~= null & k2 ~= null &
1392                    sa..contents = sb..contents & sa..size = sb..size"
1393        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1394        ensures "True" */
```

```
    {
        Object r1a = sa.put(k1, v1);
        boolean r2a = sa.containsKey(k2);
        /*: assume "~(k1 ~= k2 | r1a ~= null)" */

        boolean r2b = sb.containsKey(k2);
        Object r1b = sb.put(k1, v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void put_get_pre_s_45(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "k1 ~= k2 | (k1, v1) : sa..contents" */
        Object r1a = sa.put(k1, v1);
        Object r2a = sa.get(k2);

        Object r2b = sb.get(k2);
        Object r1b = sb.put(k1, v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void put_get_pre_c_45(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(k1 ~= k2 | (k1, v1) : sa..contents)" */
        Object r1a = sa.put(k1, v1);
        Object r2a = sa.get(k2);

        Object r2b = sb.get(k2);
        Object r1b = sb.put(k1, v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void put_get_between_s_46(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.put(k1, v1);
        /*: assume "k1 ~= k2 | r1a = v1" */
        Object r2a = sa.get(k2);

        Object r2b = sb.get(k2);
        Object r1b = sb.put(k1, v1);

```

```
        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void put_get_between_c_46(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.put(k1, v1);
        /*: assume "~(k1 ~= k2 | r1a = v1)" */
        Object r2a = sa.get(k2);

        Object r2b = sb.get(k2);
        Object r1b = sb.put(k1, v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void put_get_post_s_47(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.put(k1, v1);
        Object r2a = sa.get(k2);
        /*: assume "k1 ~= k2 | r1a = v1" */

        Object r2b = sb.get(k2);
        Object r1b = sb.put(k1, v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void put_get_post_c_47(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.put(k1, v1);
        Object r2a = sa.get(k2);
        /*: assume "~(k1 ~= k2 | r1a = v1)" */

        Object r2b = sb.get(k2);
        Object r1b = sb.put(k1, v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void put_put_pre_s_48(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null & v2 ~= null &
```

```
1504                 sa..contents = sb..contents & sa..size = sb..size"
1505         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1506         ensures "True" */
1507     {
1508         /*: assume "k1 ~= k2 | ((k1, v1) : sa..contents & v1 = v2)" */
1509         Object r1a = sa.put(k1, v1);
1510         Object r2a = sa.put(k2, v2);
1511
1512         Object r2b = sb.put(k2, v2);
1513         Object r1b = sb.put(k1, v1);
1514
1515         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
1516     }
1517
1518     static void put_put_pre_c_48(HashTable sa, HashTable sb, Object k1, Object v1,
            Object k2, Object v2)
1519     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & v1 ~= null & k2 ~= null & v2 ~= null &
1520                 sa..contents = sb..contents & sa..size = sb..size"
1521         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1522         ensures "True" */
1523     {
1524         /*: assume "~(k1 ~= k2 | ((k1, v1) : sa..contents & v1 = v2))" */
1525         Object r1a = sa.put(k1, v1);
1526         Object r2a = sa.put(k2, v2);
1527
1528         Object r2b = sb.put(k2, v2);
1529         Object r1b = sb.put(k1, v1);
1530
1531         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
1532     }
1533
1534     static void put_put_between_s_49(HashTable sa, HashTable sb, Object k1, Object v1,
            Object k2, Object v2)
1535     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & v1 ~= null & k2 ~= null & v2 ~= null &
1536                 sa..contents = sb..contents & sa..size = sb..size"
1537         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1538         ensures "True" */
1539     {
1540         Object r1a = sa.put(k1, v1);
1541         /*: assume "k1 ~= k2 | (r1a = v1 & v1 = v2)" */
1542         Object r2a = sa.put(k2, v2);
1543
1544         Object r2b = sb.put(k2, v2);
1545         Object r1b = sb.put(k1, v1);
1546
1547         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
1548     }
1549
1550     static void put_put_between_c_49(HashTable sa, HashTable sb, Object k1, Object v1,
            Object k2, Object v2)
1551     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & v1 ~= null & k2 ~= null & v2 ~= null &
1552                 sa..contents = sb..contents & sa..size = sb..size"
1553         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1554         ensures "True" */
1555     {
1556         Object r1a = sa.put(k1, v1);
1557         /*: assume "~(k1 ~= k2 | (r1a = v1 & v1 = v2))" */
1558         Object r2a = sa.put(k2, v2);
1559
```

```
1560        Object r2b = sb.put(k2, v2);
1561        Object r1b = sb.put(k1, v1);
1562
1563        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
1564    }
1565
1566    static void put_put_post_s_50(HashTable sa, HashTable sb, Object k1, Object v1,
            Object k2, Object v2)
1567    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & v1 ~= null & k2 ~= null & v2 ~= null &
1568                    sa..contents = sb..contents & sa..size = sb..size"
1569        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1570        ensures "True" */
1571    {
1572        Object r1a = sa.put(k1, v1);
1573        Object r2a = sa.put(k2, v2);
1574        /*: assume "k1 ~= k2 | (r1a = v1 & v1 = v2)" */
1575
1576        Object r2b = sb.put(k2, v2);
1577        Object r1b = sb.put(k1, v1);
1578
1579        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
1580    }
1581
1582    static void put_put_post_c_50(HashTable sa, HashTable sb, Object k1, Object v1,
            Object k2, Object v2)
1583    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & v1 ~= null & k2 ~= null & v2 ~= null &
1584                    sa..contents = sb..contents & sa..size = sb..size"
1585        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1586        ensures "True" */
1587    {
1588        Object r1a = sa.put(k1, v1);
1589        Object r2a = sa.put(k2, v2);
1590        /*: assume "~(k1 ~= k2 | (r1a = v1 & v1 = v2))" */
1591
1592        Object r2b = sb.put(k2, v2);
1593        Object r1b = sb.put(k1, v1);
1594
1595        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
1596    }
1597
1598    static void put_put_pre_s_51(HashTable sa, HashTable sb, Object k1, Object v1,
            Object k2, Object v2)
1599    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & v1 ~= null & k2 ~= null & v2 ~= null &
1600                    sa..contents = sb..contents & sa..size = sb..size"
1601        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1602        ensures "True" */
1603    {
1604        /*: assume "k1 ~= k2 | ((k1, v1) : sa..contents & v1 = v2)" */
1605        Object r1a = sa.put(k1, v1);
1606        sa.put(k2, v2);
1607
1608        sb.put(k2, v2);
1609        Object r1b = sb.put(k1, v1);
1610
1611        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1612    }
1613
1614    static void put_put_pre_c_51(HashTable sa, HashTable sb, Object k1, Object v1,
            Object k2, Object v2)
```

```
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(k1 ~= k2 | ((k1, v1) : sa..contents & v1 = v2))" */
        Object r1a = sa.put(k1, v1);
        sa.put(k2, v2);

        sb.put(k2, v2);
        Object r1b = sb.put(k1, v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void put_put_between_s_52(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.put(k1, v1);
        /*: assume "k1 ~= k2 | (r1a = v1 & v1 = v2)" */
        sa.put(k2, v2);

        sb.put(k2, v2);
        Object r1b = sb.put(k1, v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void put_put_between_c_52(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.put(k1, v1);
        /*: assume "~(k1 ~= k2 | (r1a = v1 & v1 = v2))" */
        sa.put(k2, v2);

        sb.put(k2, v2);
        Object r1b = sb.put(k1, v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void put_put_post_s_53(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.put(k1, v1);
        sa.put(k2, v2);
        /*: assume "k1 ~= k2 | (r1a = v1 & v1 = v2)" */

        sb.put(k2, v2);
```

```
1673        Object r1b = sb.put(k1, v1);
1674
1675        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1676    }
1677
1678    static void put_put_post_c_53(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2, Object v2)
1679    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null & v2 ~= null &
1680                sa..contents = sb..contents & sa..size = sb..size"
1681        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1682        ensures "True" */
1683    {
1684        Object r1a = sa.put(k1, v1);
1685        sa.put(k2, v2);
1686        /*: assume "~(k1 ~= k2 | (r1a = v1 & v1 = v2))" */
1687
1688        sb.put(k2, v2);
1689        Object r1b = sb.put(k1, v1);
1690
1691        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1692    }
1693
1694    static void put_remove_pre_s_54(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2)
1695    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
1696                sa..contents = sb..contents & sa..size = sb..size"
1697        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1698        ensures "True" */
1699    {
1700        /*: assume "k1 ~= k2" */
1701        Object r1a = sa.put(k1, v1);
1702        Object r2a = sa.remove(k2);
1703
1704        Object r2b = sb.remove(k2);
1705        Object r1b = sb.put(k1, v1);
1706
1707        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
1708    }
1709
1710    static void put_remove_pre_c_54(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2)
1711    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
1712                sa..contents = sb..contents & sa..size = sb..size"
1713        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1714        ensures "True" */
1715    {
1716        /*: assume "~(k1 ~= k2)" */
1717        Object r1a = sa.put(k1, v1);
1718        Object r2a = sa.remove(k2);
1719
1720        Object r2b = sb.remove(k2);
1721        Object r1b = sb.put(k1, v1);
1722
1723        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
1724    }
1725
1726    static void put_remove_between_s_55(HashTable sa, HashTable sb, Object k1, Object
        v1, Object k2)
1727    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
```

```
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.put(k1, v1);
        /*: assume "k1 ~= k2" */
        Object r2a = sa.remove(k2);

        Object r2b = sb.remove(k2);
        Object r1b = sb.put(k1, v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void put_remove_between_c_55(HashTable sa, HashTable sb, Object k1, Object
        v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.put(k1, v1);
        /*: assume "~(k1 ~= k2)" */
        Object r2a = sa.remove(k2);

        Object r2b = sb.remove(k2);
        Object r1b = sb.put(k1, v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void put_remove_post_s_56(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.put(k1, v1);
        Object r2a = sa.remove(k2);
        /*: assume "k1 ~= k2" */

        Object r2b = sb.remove(k2);
        Object r1b = sb.put(k1, v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void put_remove_post_c_56(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.put(k1, v1);
        Object r2a = sa.remove(k2);
        /*: assume "~(k1 ~= k2)" */

```

```
1784        Object r2b = sb.remove(k2);
1785        Object r1b = sb.put(k1, v1);
1786
1787        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
1788    }
1789
1790    static void put_remove_pre_s_57(HashTable sa, HashTable sb, Object k1, Object v1,
            Object k2)
1791    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & v1 ~= null & k2 ~= null &
1792                    sa..contents = sb..contents & sa..size = sb..size"
1793        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1794        ensures "True" */
1795    {
1796        /*: assume "k1 ~= k2" */
1797        Object r1a = sa.put(k1, v1);
1798        sa.remove(k2);
1799
1800        sb.remove(k2);
1801        Object r1b = sb.put(k1, v1);
1802
1803        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1804    }
1805
1806    static void put_remove_pre_c_57(HashTable sa, HashTable sb, Object k1, Object v1,
            Object k2)
1807    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & v1 ~= null & k2 ~= null &
1808                    sa..contents = sb..contents & sa..size = sb..size"
1809        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1810        ensures "True" */
1811    {
1812        /*: assume "~(k1 ~= k2)" */
1813        Object r1a = sa.put(k1, v1);
1814        sa.remove(k2);
1815
1816        sb.remove(k2);
1817        Object r1b = sb.put(k1, v1);
1818
1819        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1820    }
1821
1822    static void put_remove_between_s_58(HashTable sa, HashTable sb, Object k1, Object
            v1, Object k2)
1823    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & v1 ~= null & k2 ~= null &
1824                    sa..contents = sb..contents & sa..size = sb..size"
1825        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1826        ensures "True" */
1827    {
1828        Object r1a = sa.put(k1, v1);
1829        /*: assume "k1 ~= k2" */
1830        sa.remove(k2);
1831
1832        sb.remove(k2);
1833        Object r1b = sb.put(k1, v1);
1834
1835        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1836    }
1837
1838    static void put_remove_between_c_58(HashTable sa, HashTable sb, Object k1, Object
            v1, Object k2)
1839    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & v1 ~= null & k2 ~= null &
```

```
1840                 sa..contents = sb..contents & sa..size = sb..size"
1841         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1842         ensures "True" */
1843     {
1844         Object r1a = sa.put(k1, v1);
1845         /*: assume "~(k1 ~= k2)" */
1846         sa.remove(k2);
1847 
1848         sb.remove(k2);
1849         Object r1b = sb.put(k1, v1);
1850 
1851         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1852     }
1853 
1854     static void put_remove_post_s_59(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2)
1855     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
1856                 sa..contents = sb..contents & sa..size = sb..size"
1857         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1858         ensures "True" */
1859     {
1860         Object r1a = sa.put(k1, v1);
1861         sa.remove(k2);
1862         /*: assume "k1 ~= k2" */
1863 
1864         sb.remove(k2);
1865         Object r1b = sb.put(k1, v1);
1866 
1867         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1868     }
1869 
1870     static void put_remove_post_c_59(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2)
1871     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
1872                 sa..contents = sb..contents & sa..size = sb..size"
1873         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1874         ensures "True" */
1875     {
1876         Object r1a = sa.put(k1, v1);
1877         sa.remove(k2);
1878         /*: assume "~(k1 ~= k2)" */
1879 
1880         sb.remove(k2);
1881         Object r1b = sb.put(k1, v1);
1882 
1883         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1884     }
1885 
1886     static void put_size_pre_s_60(HashTable sa, HashTable sb, Object k1, Object v1)
1887     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null &
1888                 sa..contents = sb..contents & sa..size = sb..size"
1889         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1890         ensures "True" */
1891     {
1892         /*: assume "EX v. (k1, v) : sa..contents" */
1893         Object r1a = sa.put(k1, v1);
1894         int r2a = sa.size();
1895 
1896         int r2b = sb.size();
1897         Object r1b = sb.put(k1, v1);
1898 
```

```
        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void put_size_pre_c_60(HashTable sa, HashTable sb, Object k1, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(EX v. (k1, v) : sa..contents)" */
        Object r1a = sa.put(k1, v1);
        int r2a = sa.size();

        int r2b = sb.size();
        Object r1b = sb.put(k1, v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void put_size_between_s_61(HashTable sa, HashTable sb, Object k1, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.put(k1, v1);
        /*: assume "r1a ~= null" */
        int r2a = sa.size();

        int r2b = sb.size();
        Object r1b = sb.put(k1, v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void put_size_between_c_61(HashTable sa, HashTable sb, Object k1, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.put(k1, v1);
        /*: assume "~(r1a ~= null)" */
        int r2a = sa.size();

        int r2b = sb.size();
        Object r1b = sb.put(k1, v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void put_size_post_s_62(HashTable sa, HashTable sb, Object k1, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
```

```
        Object r1a = sa.put(k1, v1);
        int r2a = sa.size();
        /*: assume "r1a ~= null" */

        int r2b = sb.size();
        Object r1b = sb.put(k1, v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void put_size_post_c_62(HashTable sa, HashTable sb, Object k1, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.put(k1, v1);
        int r2a = sa.size();
        /*: assume "~(r1a ~= null)" */

        int r2b = sb.size();
        Object r1b = sb.put(k1, v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void put_containsKey_pre_s_63(HashTable sa, HashTable sb, Object k1, Object
        v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "k1 ~= k2 | (EX v. (k1, v) : sa..contents)" */
        sa.put(k1, v1);
        boolean r2a = sa.containsKey(k2);

        boolean r2b = sb.containsKey(k2);
        sb.put(k1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void put_containsKey_pre_c_63(HashTable sa, HashTable sb, Object k1, Object
        v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(k1 ~= k2 | (EX v. (k1, v) : sa..contents))" */
        sa.put(k1, v1);
        boolean r2a = sa.containsKey(k2);

        boolean r2b = sb.containsKey(k2);
        sb.put(k1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }
```

```
    static void put_containsKey_between_s_64(HashTable sa, HashTable sb, Object k1,
        Object v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.put(k1, v1);
        /*: assume "k1 ~= k2 | (EX v. (k1, v) : sa..(old contents))" */
        boolean r2a = sa.containsKey(k2);

        boolean r2b = sb.containsKey(k2);
        sb.put(k1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void put_containsKey_between_c_64(HashTable sa, HashTable sb, Object k1,
        Object v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.put(k1, v1);
        /*: assume "~(k1 ~= k2 | (EX v. (k1, v) : sa..(old contents)))" */
        boolean r2a = sa.containsKey(k2);

        boolean r2b = sb.containsKey(k2);
        sb.put(k1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void put_containsKey_post_s_65(HashTable sa, HashTable sb, Object k1, Object
        v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.put(k1, v1);
        boolean r2a = sa.containsKey(k2);
        /*: assume "k1 ~= k2 | (EX v. (k1, v) : sa..(old contents))" */

        boolean r2b = sb.containsKey(k2);
        sb.put(k1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void put_containsKey_post_c_65(HashTable sa, HashTable sb, Object k1, Object
        v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.put(k1, v1);
        boolean r2a = sa.containsKey(k2);
        /*: assume "~(k1 ~= k2 | (EX v. (k1, v) : sa..(old contents)))" */
```

```
2071
2072        boolean r2b = sb.containsKey(k2);
2073        sb.put(k1, v1);
2074
2075        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2076    }
2077
2078    static void put_get_pre_s_66(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2)
2079    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
2080                sa..contents = sb..contents & sa..size = sb..size"
2081        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2082        ensures "True" */
2083    {
2084        /*: assume "k1 ~= k2 | (k1, v1) : sa..contents" */
2085        sa.put(k1, v1);
2086        Object r2a = sa.get(k2);
2087
2088        Object r2b = sb.get(k2);
2089        sb.put(k1, v1);
2090
2091        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2092    }
2093
2094    static void put_get_pre_c_66(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2)
2095    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
2096                sa..contents = sb..contents & sa..size = sb..size"
2097        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2098        ensures "True" */
2099    {
2100        /*: assume "~(k1 ~= k2 | (k1, v1) : sa..contents)" */
2101        sa.put(k1, v1);
2102        Object r2a = sa.get(k2);
2103
2104        Object r2b = sb.get(k2);
2105        sb.put(k1, v1);
2106
2107        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2108    }
2109
2110    static void put_get_between_s_67(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2)
2111    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
2112                sa..contents = sb..contents & sa..size = sb..size"
2113        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2114        ensures "True" */
2115    {
2116        sa.put(k1, v1);
2117        /*: assume "k1 ~= k2 | (k1, v1) : sa..(old contents)" */
2118        Object r2a = sa.get(k2);
2119
2120        Object r2b = sb.get(k2);
2121        sb.put(k1, v1);
2122
2123        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2124    }
2125
2126    static void put_get_between_c_67(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2)
2127    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
```

```
2128                 sa..contents = sb..contents & sa..size = sb..size"
2129         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2130         ensures "True" */
2131     {
2132         sa.put(k1, v1);
2133         /*: assume "~(k1 ~= k2 | (k1, v1) : sa..(old contents))" */
2134         Object r2a = sa.get(k2);
2135 
2136         Object r2b = sb.get(k2);
2137         sb.put(k1, v1);
2138 
2139         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2140     }
2141 
2142     static void put_get_post_s_68(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2)
2143     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
2144                 sa..contents = sb..contents & sa..size = sb..size"
2145         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2146         ensures "True" */
2147     {
2148         sa.put(k1, v1);
2149         Object r2a = sa.get(k2);
2150         /*: assume "k1 ~= k2 | (k1, v1) : sa..(old contents)" */
2151 
2152         Object r2b = sb.get(k2);
2153         sb.put(k1, v1);
2154 
2155         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2156     }
2157 
2158     static void put_get_post_c_68(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2)
2159     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
2160                 sa..contents = sb..contents & sa..size = sb..size"
2161         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2162         ensures "True" */
2163     {
2164         sa.put(k1, v1);
2165         Object r2a = sa.get(k2);
2166         /*: assume "~(k1 ~= k2 | (k1, v1) : sa..(old contents))" */
2167 
2168         Object r2b = sb.get(k2);
2169         sb.put(k1, v1);
2170 
2171         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2172     }
2173 
2174     static void put_put_pre_s_69(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2, Object v2)
2175     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null & v2 ~= null &
2176                 sa..contents = sb..contents & sa..size = sb..size"
2177         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2178         ensures "True" */
2179     {
2180         /*: assume "k1 ~= k2 | ((k1, v1) : sa..contents & v1 = v2)" */
2181         sa.put(k1, v1);
2182         Object r2a = sa.put(k2, v2);
2183 
2184         Object r2b = sb.put(k2, v2);
2185         sb.put(k1, v1);
2186 
```

```
        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void put_put_pre_c_69(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(k1 ~= k2 | ((k1, v1) : sa..contents & v1 = v2))" */
        sa.put(k1, v1);
        Object r2a = sa.put(k2, v2);

        Object r2b = sb.put(k2, v2);
        sb.put(k1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void put_put_between_s_70(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.put(k1, v1);
        /*: assume "k1 ~= k2 | ((k1, v1) : sa..(old contents) & v1 = v2)" */
        Object r2a = sa.put(k2, v2);

        Object r2b = sb.put(k2, v2);
        sb.put(k1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void put_put_between_c_70(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.put(k1, v1);
        /*: assume "~(k1 ~= k2 | ((k1, v1) : sa..(old contents) & v1 = v2))" */
        Object r2a = sa.put(k2, v2);

        Object r2b = sb.put(k2, v2);
        sb.put(k1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void put_put_post_s_71(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
```

```
2244        sa.put(k1, v1);
2245        Object r2a = sa.put(k2, v2);
2246        /*: assume "k1 ~= k2 | ((k1, v1) : sa..(old contents) & v1 = v2)" */
2247
2248        Object r2b = sb.put(k2, v2);
2249        sb.put(k1, v1);
2250
2251        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2252    }
2253
2254    static void put_put_post_c_71(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2, Object v2)
2255    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null & v2 ~= null &
2256                sa..contents = sb..contents & sa..size = sb..size"
2257        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2258        ensures "True" */
2259    {
2260        sa.put(k1, v1);
2261        Object r2a = sa.put(k2, v2);
2262        /*: assume "~(k1 ~= k2 | ((k1, v1) : sa..(old contents) & v1 = v2))" */
2263
2264        Object r2b = sb.put(k2, v2);
2265        sb.put(k1, v1);
2266
2267        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2268    }
2269
2270    static void put_put_pre_s_72(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2, Object v2)
2271    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null & v2 ~= null &
2272                sa..contents = sb..contents & sa..size = sb..size"
2273        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2274        ensures "True" */
2275    {
2276        /*: assume "k1 ~= k2 | v1 = v2" */
2277        sa.put(k1, v1);
2278        sa.put(k2, v2);
2279
2280        sb.put(k2, v2);
2281        sb.put(k1, v1);
2282
2283        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
2284    }
2285
2286    static void put_put_pre_c_72(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2, Object v2)
2287    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null & v2 ~= null &
2288                sa..contents = sb..contents & sa..size = sb..size"
2289        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2290        ensures "True" */
2291    {
2292        /*: assume "~(k1 ~= k2 | v1 = v2)" */
2293        sa.put(k1, v1);
2294        sa.put(k2, v2);
2295
2296        sb.put(k2, v2);
2297        sb.put(k1, v1);
2298
2299        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
2300    }
2301
```

```
    static void put_put_between_s_73(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.put(k1, v1);
        /*: assume "k1 ~= k2 | v1 = v2" */
        sa.put(k2, v2);

        sb.put(k2, v2);
        sb.put(k1, v1);

        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void put_put_between_c_73(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.put(k1, v1);
        /*: assume "~(k1 ~= k2 | v1 = v2)" */
        sa.put(k2, v2);

        sb.put(k2, v2);
        sb.put(k1, v1);

        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void put_put_post_s_74(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.put(k1, v1);
        sa.put(k2, v2);
        /*: assume "k1 ~= k2 | v1 = v2" */

        sb.put(k2, v2);
        sb.put(k1, v1);

        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void put_put_post_c_74(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.put(k1, v1);
        sa.put(k2, v2);
        /*: assume "~(k1 ~= k2 | v1 = v2)" */
```

```
2359
2360        sb.put(k2, v2);
2361        sb.put(k1, v1);
2362
2363        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
2364    }
2365
2366    static void put_remove_pre_s_75(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2)
2367    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
2368                sa..contents = sb..contents & sa..size = sb..size"
2369        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2370        ensures "True" */
2371    {
2372        /*: assume "k1 ~= k2" */
2373        sa.put(k1, v1);
2374        Object r2a = sa.remove(k2);
2375
2376        Object r2b = sb.remove(k2);
2377        sb.put(k1, v1);
2378
2379        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2380    }
2381
2382    static void put_remove_pre_c_75(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2)
2383    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
2384                sa..contents = sb..contents & sa..size = sb..size"
2385        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2386        ensures "True" */
2387    {
2388        /*: assume "~(k1 ~= k2)" */
2389        sa.put(k1, v1);
2390        Object r2a = sa.remove(k2);
2391
2392        Object r2b = sb.remove(k2);
2393        sb.put(k1, v1);
2394
2395        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2396    }
2397
2398    static void put_remove_between_s_76(HashTable sa, HashTable sb, Object k1, Object
        v1, Object k2)
2399    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
2400                sa..contents = sb..contents & sa..size = sb..size"
2401        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2402        ensures "True" */
2403    {
2404        sa.put(k1, v1);
2405        /*: assume "k1 ~= k2" */
2406        Object r2a = sa.remove(k2);
2407
2408        Object r2b = sb.remove(k2);
2409        sb.put(k1, v1);
2410
2411        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2412    }
2413
2414    static void put_remove_between_c_76(HashTable sa, HashTable sb, Object k1, Object
        v1, Object k2)
2415    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
```

```
2416                 sa..contents = sb..contents & sa..size = sb..size"
2417         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2418         ensures "True" */
2419     {
2420         sa.put(k1, v1);
2421         /*: assume "~(k1 ~= k2)" */
2422         Object r2a = sa.remove(k2);
2423 
2424         Object r2b = sb.remove(k2);
2425         sb.put(k1, v1);
2426 
2427         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2428     }
2429 
2430     static void put_remove_post_s_77(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2)
2431     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
2432                 sa..contents = sb..contents & sa..size = sb..size"
2433         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2434         ensures "True" */
2435     {
2436         sa.put(k1, v1);
2437         Object r2a = sa.remove(k2);
2438         /*: assume "k1 ~= k2" */
2439 
2440         Object r2b = sb.remove(k2);
2441         sb.put(k1, v1);
2442 
2443         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2444     }
2445 
2446     static void put_remove_post_c_77(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2)
2447     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
2448                 sa..contents = sb..contents & sa..size = sb..size"
2449         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2450         ensures "True" */
2451     {
2452         sa.put(k1, v1);
2453         Object r2a = sa.remove(k2);
2454         /*: assume "~(k1 ~= k2)" */
2455 
2456         Object r2b = sb.remove(k2);
2457         sb.put(k1, v1);
2458 
2459         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2460     }
2461 
2462     static void put_remove_pre_s_78(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2)
2463     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
2464                 sa..contents = sb..contents & sa..size = sb..size"
2465         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2466         ensures "True" */
2467     {
2468         /*: assume "k1 ~= k2" */
2469         sa.put(k1, v1);
2470         sa.remove(k2);
2471 
2472         sb.remove(k2);
2473         sb.put(k1, v1);
2474 
```

```
        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void put_remove_pre_c_78(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(k1 ~= k2)" */
        sa.put(k1, v1);
        sa.remove(k2);

        sb.remove(k2);
        sb.put(k1, v1);

        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void put_remove_between_s_79(HashTable sa, HashTable sb, Object k1, Object
        v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.put(k1, v1);
        /*: assume "k1 ~= k2" */
        sa.remove(k2);

        sb.remove(k2);
        sb.put(k1, v1);

        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void put_remove_between_c_79(HashTable sa, HashTable sb, Object k1, Object
        v1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.put(k1, v1);
        /*: assume "~(k1 ~= k2)" */
        sa.remove(k2);

        sb.remove(k2);
        sb.put(k1, v1);

        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void put_remove_post_s_80(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
```

```
2532        sa.put(k1, v1);
2533        sa.remove(k2);
2534        /*: assume "k1 ~= k2" */
2535
2536        sb.remove(k2);
2537        sb.put(k1, v1);
2538
2539        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
2540    }
2541
2542    static void put_remove_post_c_80(HashTable sa, HashTable sb, Object k1, Object v1,
        Object k2)
2543    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null & k2 ~= null &
2544                sa..contents = sb..contents & sa..size = sb..size"
2545        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2546        ensures "True" */
2547    {
2548        sa.put(k1, v1);
2549        sa.remove(k2);
2550        /*: assume "~(k1 ~= k2)" */
2551
2552        sb.remove(k2);
2553        sb.put(k1, v1);
2554
2555        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
2556    }
2557
2558    static void put_size_pre_s_81(HashTable sa, HashTable sb, Object k1, Object v1)
2559    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null &
2560                sa..contents = sb..contents & sa..size = sb..size"
2561        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2562        ensures "True" */
2563    {
2564        /*: assume "EX v. (k1, v) : sa..contents" */
2565        sa.put(k1, v1);
2566        int r2a = sa.size();
2567
2568        int r2b = sb.size();
2569        sb.put(k1, v1);
2570
2571        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2572    }
2573
2574    static void put_size_pre_c_81(HashTable sa, HashTable sb, Object k1, Object v1)
2575    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null &
2576                sa..contents = sb..contents & sa..size = sb..size"
2577        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2578        ensures "True" */
2579    {
2580        /*: assume "~(EX v. (k1, v) : sa..contents)" */
2581        sa.put(k1, v1);
2582        int r2a = sa.size();
2583
2584        int r2b = sb.size();
2585        sb.put(k1, v1);
2586
2587        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2588    }
2589
2590    static void put_size_between_s_82(HashTable sa, HashTable sb, Object k1, Object v1)
2591    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null &
```

```
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.put(k1, v1);
        /*: assume "EX v. (k1, v) : sa..(old contents)" */
        int r2a = sa.size();

        int r2b = sb.size();
        sb.put(k1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void put_size_between_c_82(HashTable sa, HashTable sb, Object k1, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.put(k1, v1);
        /*: assume "~(EX v. (k1, v) : sa..(old contents))" */
        int r2a = sa.size();

        int r2b = sb.size();
        sb.put(k1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void put_size_post_s_83(HashTable sa, HashTable sb, Object k1, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.put(k1, v1);
        int r2a = sa.size();
        /*: assume "EX v. (k1, v) : sa..(old contents)" */

        int r2b = sb.size();
        sb.put(k1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void put_size_post_c_83(HashTable sa, HashTable sb, Object k1, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & v1 ~= null &
                sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.put(k1, v1);
        int r2a = sa.size();
        /*: assume "~(EX v. (k1, v) : sa..(old contents))" */

        int r2b = sb.size();
        sb.put(k1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }
```

```
2654    static void remove_containsKey_pre_s_84(HashTable sa, HashTable sb, Object k1,
            Object k2)
2655    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
2656                sa..contents = sb..contents & sa..size = sb..size"
2657        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2658        ensures "True" */
2659    {
2660        /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
2661        Object r1a = sa.remove(k1);
2662        boolean r2a = sa.containsKey(k2);
2663
2664        boolean r2b = sb.containsKey(k2);
2665        Object r1b = sb.remove(k1);
2666
2667        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
2668    }
2669
2670    static void remove_containsKey_pre_c_84(HashTable sa, HashTable sb, Object k1,
            Object k2)
2671    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
2672                sa..contents = sb..contents & sa..size = sb..size"
2673        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2674        ensures "True" */
2675    {
2676        /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
2677        Object r1a = sa.remove(k1);
2678        boolean r2a = sa.containsKey(k2);
2679
2680        boolean r2b = sb.containsKey(k2);
2681        Object r1b = sb.remove(k1);
2682
2683        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
2684    }
2685
2686    static void remove_containsKey_between_s_85(HashTable sa, HashTable sb, Object k1,
            Object k2)
2687    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
2688                sa..contents = sb..contents & sa..size = sb..size"
2689        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2690        ensures "True" */
2691    {
2692        Object r1a = sa.remove(k1);
2693        /*: assume "k1 ~= k2 | r1a = null" */
2694        boolean r2a = sa.containsKey(k2);
2695
2696        boolean r2b = sb.containsKey(k2);
2697        Object r1b = sb.remove(k1);
2698
2699        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
2700    }
2701
2702    static void remove_containsKey_between_c_85(HashTable sa, HashTable sb, Object k1,
            Object k2)
2703    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
2704                sa..contents = sb..contents & sa..size = sb..size"
2705        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2706        ensures "True" */
2707    {
```

```
2708        Object r1a = sa.remove(k1);
2709        /*: assume "~(k1 ~= k2 | r1a = null)" */
2710        boolean r2a = sa.containsKey(k2);
2711
2712        boolean r2b = sb.containsKey(k2);
2713        Object r1b = sb.remove(k1);
2714
2715        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
2716    }
2717
2718    static void remove_containsKey_post_s_86(HashTable sa, HashTable sb, Object k1,
        Object k2)
2719    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
2720                sa..contents = sb..contents & sa..size = sb..size"
2721        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2722        ensures "True" */
2723    {
2724        Object r1a = sa.remove(k1);
2725        boolean r2a = sa.containsKey(k2);
2726        /*: assume "k1 ~= k2 | r1a = null" */
2727
2728        boolean r2b = sb.containsKey(k2);
2729        Object r1b = sb.remove(k1);
2730
2731        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
2732    }
2733
2734    static void remove_containsKey_post_c_86(HashTable sa, HashTable sb, Object k1,
        Object k2)
2735    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
2736                sa..contents = sb..contents & sa..size = sb..size"
2737        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2738        ensures "True" */
2739    {
2740        Object r1a = sa.remove(k1);
2741        boolean r2a = sa.containsKey(k2);
2742        /*: assume "~(k1 ~= k2 | r1a = null)" */
2743
2744        boolean r2b = sb.containsKey(k2);
2745        Object r1b = sb.remove(k1);
2746
2747        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
2748    }
2749
2750    static void remove_get_pre_s_87(HashTable sa, HashTable sb, Object k1, Object k2)
2751    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
2752                sa..contents = sb..contents & sa..size = sb..size"
2753        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2754        ensures "True" */
2755    {
2756        /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
2757        Object r1a = sa.remove(k1);
2758        Object r2a = sa.get(k2);
2759
2760        Object r2b = sb.get(k2);
2761        Object r1b = sb.remove(k1);
2762
2763        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
```

```
    }

    static void remove_get_pre_c_87(HashTable sa, HashTable sb, Object k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
         ensures "True" */
    {
        /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
        Object r1a = sa.remove(k1);
        Object r2a = sa.get(k2);

        Object r2b = sb.get(k2);
        Object r1b = sb.remove(k1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void remove_get_between_s_88(HashTable sa, HashTable sb, Object k1, Object
        k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
         ensures "True" */
    {
        Object r1a = sa.remove(k1);
        /*: assume "k1 ~= k2 | r1a = null" */
        Object r2a = sa.get(k2);

        Object r2b = sb.get(k2);
        Object r1b = sb.remove(k1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void remove_get_between_c_88(HashTable sa, HashTable sb, Object k1, Object
        k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
         ensures "True" */
    {
        Object r1a = sa.remove(k1);
        /*: assume "~(k1 ~= k2 | r1a = null)" */
        Object r2a = sa.get(k2);

        Object r2b = sb.get(k2);
        Object r1b = sb.remove(k1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void remove_get_post_s_89(HashTable sa, HashTable sb, Object k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
         ensures "True" */
    {
```

```
2820        Object r1a = sa.remove(k1);
2821        Object r2a = sa.get(k2);
2822        /*: assume "k1 ~= k2 | r1a = null" */
2823
2824        Object r2b = sb.get(k2);
2825        Object r1b = sb.remove(k1);
2826
2827        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
2828    }
2829
2830    static void remove_get_post_c_89(HashTable sa, HashTable sb, Object k1, Object k2)
2831    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
2832                    sa..contents = sb..contents & sa..size = sb..size"
2833        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2834        ensures "True" */
2835    {
2836        Object r1a = sa.remove(k1);
2837        Object r2a = sa.get(k2);
2838        /*: assume "~(k1 ~= k2 | r1a = null)" */
2839
2840        Object r2b = sb.get(k2);
2841        Object r1b = sb.remove(k1);
2842
2843        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
2844    }
2845
2846    static void remove_put_pre_s_90(HashTable sa, HashTable sb, Object k1, Object k2,
            Object v2)
2847    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null & v2 ~= null &
2848                    sa..contents = sb..contents & sa..size = sb..size"
2849        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2850        ensures "True" */
2851    {
2852        /*: assume "k1 ~= k2" */
2853        Object r1a = sa.remove(k1);
2854        Object r2a = sa.put(k2, v2);
2855
2856        Object r2b = sb.put(k2, v2);
2857        Object r1b = sb.remove(k1);
2858
2859        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
2860    }
2861
2862    static void remove_put_pre_c_90(HashTable sa, HashTable sb, Object k1, Object k2,
            Object v2)
2863    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null & v2 ~= null &
2864                    sa..contents = sb..contents & sa..size = sb..size"
2865        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2866        ensures "True" */
2867    {
2868        /*: assume "~(k1 ~= k2)" */
2869        Object r1a = sa.remove(k1);
2870        Object r2a = sa.put(k2, v2);
2871
2872        Object r2b = sb.put(k2, v2);
2873        Object r1b = sb.remove(k1);
2874
2875        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
```

```
2876    }
2877
2878    static void remove_put_between_s_91(HashTable sa, HashTable sb, Object k1, Object
            k2, Object v2)
2879    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null & v2 ~= null &
2880                sa..contents = sb..contents & sa..size = sb..size"
2881        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2882        ensures "True" */
2883    {
2884        Object r1a = sa.remove(k1);
2885        /*: assume "k1 ~= k2" */
2886        Object r2a = sa.put(k2, v2);
2887
2888        Object r2b = sb.put(k2, v2);
2889        Object r1b = sb.remove(k1);
2890
2891        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
2892    }
2893
2894    static void remove_put_between_c_91(HashTable sa, HashTable sb, Object k1, Object
            k2, Object v2)
2895    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null & v2 ~= null &
2896                sa..contents = sb..contents & sa..size = sb..size"
2897        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2898        ensures "True" */
2899    {
2900        Object r1a = sa.remove(k1);
2901        /*: assume "~(k1 ~= k2)" */
2902        Object r2a = sa.put(k2, v2);
2903
2904        Object r2b = sb.put(k2, v2);
2905        Object r1b = sb.remove(k1);
2906
2907        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
2908    }
2909
2910    static void remove_put_post_s_92(HashTable sa, HashTable sb, Object k1, Object k2,
            Object v2)
2911    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null & v2 ~= null &
2912                sa..contents = sb..contents & sa..size = sb..size"
2913        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2914        ensures "True" */
2915    {
2916        Object r1a = sa.remove(k1);
2917        Object r2a = sa.put(k2, v2);
2918        /*: assume "k1 ~= k2" */
2919
2920        Object r2b = sb.put(k2, v2);
2921        Object r1b = sb.remove(k1);
2922
2923        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
2924    }
2925
2926    static void remove_put_post_c_92(HashTable sa, HashTable sb, Object k1, Object k2,
            Object v2)
2927    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null & v2 ~= null &
2928                sa..contents = sb..contents & sa..size = sb..size"
2929        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
```

```
2930        ensures "True" */
2931    {
2932        Object r1a = sa.remove(k1);
2933        Object r2a = sa.put(k2, v2);
2934        /*: assume "~(k1 ~= k2)" */
2935
2936        Object r2b = sb.put(k2, v2);
2937        Object r1b = sb.remove(k1);
2938
2939        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
2940    }
2941
2942    static void remove_put_pre_s_93(HashTable sa, HashTable sb, Object k1, Object k2,
        Object v2)
2943    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null & v2 ~= null &
2944                    sa..contents = sb..contents & sa..size = sb..size"
2945        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2946        ensures "True" */
2947    {
2948        /*: assume "k1 ~= k2" */
2949        Object r1a = sa.remove(k1);
2950        sa.put(k2, v2);
2951
2952        sb.put(k2, v2);
2953        Object r1b = sb.remove(k1);
2954
2955        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
2956    }
2957
2958    static void remove_put_pre_c_93(HashTable sa, HashTable sb, Object k1, Object k2,
        Object v2)
2959    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null & v2 ~= null &
2960                    sa..contents = sb..contents & sa..size = sb..size"
2961        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2962        ensures "True" */
2963    {
2964        /*: assume "~(k1 ~= k2)" */
2965        Object r1a = sa.remove(k1);
2966        sa.put(k2, v2);
2967
2968        sb.put(k2, v2);
2969        Object r1b = sb.remove(k1);
2970
2971        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
2972    }
2973
2974    static void remove_put_between_s_94(HashTable sa, HashTable sb, Object k1, Object
        k2, Object v2)
2975    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null & v2 ~= null &
2976                    sa..contents = sb..contents & sa..size = sb..size"
2977        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2978        ensures "True" */
2979    {
2980        Object r1a = sa.remove(k1);
2981        /*: assume "k1 ~= k2" */
2982        sa.put(k2, v2);
2983
2984        sb.put(k2, v2);
2985        Object r1b = sb.remove(k1);
2986
2987        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
```

```
2988    }
2989
2990    static void remove_put_between_c_94(HashTable sa, HashTable sb, Object k1, Object
            k2, Object v2)
2991    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null & v2 ~= null &
2992                sa..contents = sb..contents & sa..size = sb..size"
2993        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2994        ensures "True" */
2995    {
2996        Object r1a = sa.remove(k1);
2997        /*: assume "~(k1 ~= k2)" */
2998        sa.put(k2, v2);
2999
3000        sb.put(k2, v2);
3001        Object r1b = sb.remove(k1);
3002
3003        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
3004    }
3005
3006    static void remove_put_post_s_95(HashTable sa, HashTable sb, Object k1, Object k2,
            Object v2)
3007    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null & v2 ~= null &
3008                sa..contents = sb..contents & sa..size = sb..size"
3009        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3010        ensures "True" */
3011    {
3012        Object r1a = sa.remove(k1);
3013        sa.put(k2, v2);
3014        /*: assume "k1 ~= k2" */
3015
3016        sb.put(k2, v2);
3017        Object r1b = sb.remove(k1);
3018
3019        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
3020    }
3021
3022    static void remove_put_post_c_95(HashTable sa, HashTable sb, Object k1, Object k2,
            Object v2)
3023    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null & v2 ~= null &
3024                sa..contents = sb..contents & sa..size = sb..size"
3025        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3026        ensures "True" */
3027    {
3028        Object r1a = sa.remove(k1);
3029        sa.put(k2, v2);
3030        /*: assume "~(k1 ~= k2)" */
3031
3032        sb.put(k2, v2);
3033        Object r1b = sb.remove(k1);
3034
3035        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
3036    }
3037
3038    static void remove_remove_pre_s_96(HashTable sa, HashTable sb, Object k1, Object k2)
3039    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
3040                sa..contents = sb..contents & sa..size = sb..size"
3041        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3042        ensures "True" */
3043    {
3044        /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
3045        Object r1a = sa.remove(k1);
```

```
        Object r2a = sa.remove(k2);

        Object r2b = sb.remove(k2);
        Object r1b = sb.remove(k1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void remove_remove_pre_c_96(HashTable sa, HashTable sb, Object k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
        Object r1a = sa.remove(k1);
        Object r2a = sa.remove(k2);

        Object r2b = sb.remove(k2);
        Object r1b = sb.remove(k1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void remove_remove_between_s_97(HashTable sa, HashTable sb, Object k1,
        Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.remove(k1);
        /*: assume "k1 ~= k2 | r1a = null" */
        Object r2a = sa.remove(k2);

        Object r2b = sb.remove(k2);
        Object r1b = sb.remove(k1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void remove_remove_between_c_97(HashTable sa, HashTable sb, Object k1,
        Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.remove(k1);
        /*: assume "~(k1 ~= k2 | r1a = null)" */
        Object r2a = sa.remove(k2);

        Object r2b = sb.remove(k2);
        Object r1b = sb.remove(k1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }
```

```
3102    static void remove_remove_post_s_98(HashTable sa, HashTable sb, Object k1, Object
            k2)
3103    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
3104                    sa..contents = sb..contents & sa..size = sb..size"
3105        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3106        ensures "True" */
3107    {
3108        Object r1a = sa.remove(k1);
3109        Object r2a = sa.remove(k2);
3110        /*: assume "k1 ~= k2 | r1a = null" */
3111
3112        Object r2b = sb.remove(k2);
3113        Object r1b = sb.remove(k1);
3114
3115        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
3116    }
3117
3118    static void remove_remove_post_c_98(HashTable sa, HashTable sb, Object k1, Object
            k2)
3119    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
3120                    sa..contents = sb..contents & sa..size = sb..size"
3121        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3122        ensures "True" */
3123    {
3124        Object r1a = sa.remove(k1);
3125        Object r2a = sa.remove(k2);
3126        /*: assume "~(k1 ~= k2 | r1a = null)" */
3127
3128        Object r2b = sb.remove(k2);
3129        Object r1b = sb.remove(k1);
3130
3131        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
3132    }
3133
3134    static void remove_remove_pre_s_99(HashTable sa, HashTable sb, Object k1, Object k2)
3135    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
3136                    sa..contents = sb..contents & sa..size = sb..size"
3137        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3138        ensures "True" */
3139    {
3140        /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
3141        Object r1a = sa.remove(k1);
3142        sa.remove(k2);
3143
3144        sb.remove(k2);
3145        Object r1b = sb.remove(k1);
3146
3147        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
3148    }
3149
3150    static void remove_remove_pre_c_99(HashTable sa, HashTable sb, Object k1, Object k2)
3151    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
3152                    sa..contents = sb..contents & sa..size = sb..size"
3153        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3154        ensures "True" */
3155    {
3156        /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
3157        Object r1a = sa.remove(k1);
3158        sa.remove(k2);
```

```
3159
3160        sb.remove(k2);
3161        Object r1b = sb.remove(k1);
3162
3163        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
3164    }
3165
3166    static void remove_remove_between_s_100(HashTable sa, HashTable sb, Object k1,
        Object k2)
3167    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
3168                sa..contents = sb..contents & sa..size = sb..size"
3169        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3170        ensures "True" */
3171    {
3172        Object r1a = sa.remove(k1);
3173        /*: assume "k1 ~= k2 | r1a = null" */
3174        sa.remove(k2);
3175
3176        sb.remove(k2);
3177        Object r1b = sb.remove(k1);
3178
3179        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
3180    }
3181
3182    static void remove_remove_between_c_100(HashTable sa, HashTable sb, Object k1,
        Object k2)
3183    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
3184                sa..contents = sb..contents & sa..size = sb..size"
3185        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3186        ensures "True" */
3187    {
3188        Object r1a = sa.remove(k1);
3189        /*: assume "~(k1 ~= k2 | r1a = null)" */
3190        sa.remove(k2);
3191
3192        sb.remove(k2);
3193        Object r1b = sb.remove(k1);
3194
3195        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
3196    }
3197
3198    static void remove_remove_post_s_101(HashTable sa, HashTable sb, Object k1, Object
        k2)
3199    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
3200                sa..contents = sb..contents & sa..size = sb..size"
3201        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3202        ensures "True" */
3203    {
3204        Object r1a = sa.remove(k1);
3205        sa.remove(k2);
3206        /*: assume "k1 ~= k2 | r1a = null" */
3207
3208        sb.remove(k2);
3209        Object r1b = sb.remove(k1);
3210
3211        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
3212    }
3213
3214    static void remove_remove_post_c_101(HashTable sa, HashTable sb, Object k1, Object
        k2)
3215    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
```

```
3216                 sa..contents = sb..contents & sa..size = sb..size"
3217         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3218         ensures "True" */
3219     {
3220         Object r1a = sa.remove(k1);
3221         sa.remove(k2);
3222         /*: assume "~(k1 ~= k2 | r1a = null)" */
3223 
3224         sb.remove(k2);
3225         Object r1b = sb.remove(k1);
3226 
3227         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
3228     }
3229 
3230     static void remove_size_pre_s_102(HashTable sa, HashTable sb, Object k1)
3231     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            &
3232                 sa..contents = sb..contents & sa..size = sb..size"
3233         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3234         ensures "True" */
3235     {
3236         /*: assume "~(EX v. (k1, v) : sa..contents)" */
3237         Object r1a = sa.remove(k1);
3238         int r2a = sa.size();
3239 
3240         int r2b = sb.size();
3241         Object r1b = sb.remove(k1);
3242 
3243         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
3244     }
3245 
3246     static void remove_size_pre_c_102(HashTable sa, HashTable sb, Object k1)
3247     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            &
3248                 sa..contents = sb..contents & sa..size = sb..size"
3249         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3250         ensures "True" */
3251     {
3252         /*: assume "~(~(EX v. (k1, v) : sa..contents))" */
3253         Object r1a = sa.remove(k1);
3254         int r2a = sa.size();
3255 
3256         int r2b = sb.size();
3257         Object r1b = sb.remove(k1);
3258 
3259         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
3260     }
3261 
3262     static void remove_size_between_s_103(HashTable sa, HashTable sb, Object k1)
3263     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            &
3264                 sa..contents = sb..contents & sa..size = sb..size"
3265         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3266         ensures "True" */
3267     {
3268         Object r1a = sa.remove(k1);
3269         /*: assume "r1a = null" */
3270         int r2a = sa.size();
3271 
3272         int r2b = sb.size();
3273         Object r1b = sb.remove(k1);
3274 
```

```
        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void remove_size_between_c_103(HashTable sa, HashTable sb, Object k1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.remove(k1);
        /*: assume "~(r1a = null)" */
        int r2a = sa.size();

        int r2b = sb.size();
        Object r1b = sb.remove(k1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void remove_size_post_s_104(HashTable sa, HashTable sb, Object k1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.remove(k1);
        int r2a = sa.size();
        /*: assume "r1a = null" */

        int r2b = sb.size();
        Object r1b = sb.remove(k1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void remove_size_post_c_104(HashTable sa, HashTable sb, Object k1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        Object r1a = sa.remove(k1);
        int r2a = sa.size();
        /*: assume "~(r1a = null)" */

        int r2b = sb.size();
        Object r1b = sb.remove(k1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void remove_containsKey_pre_s_105(HashTable sa, HashTable sb, Object k1,
        Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
```

```
3331    {
3332        /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
3333        sa.remove(k1);
3334        boolean r2a = sa.containsKey(k2);
3335
3336        boolean r2b = sb.containsKey(k2);
3337        sb.remove(k1);
3338
3339        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3340    }
3341
3342    static void remove_containsKey_pre_c_105(HashTable sa, HashTable sb, Object k1,
        Object k2)
3343    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
3344                sa..contents = sb..contents & sa..size = sb..size"
3345        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3346        ensures "True" */
3347    {
3348        /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
3349        sa.remove(k1);
3350        boolean r2a = sa.containsKey(k2);
3351
3352        boolean r2b = sb.containsKey(k2);
3353        sb.remove(k1);
3354
3355        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3356    }
3357
3358    static void remove_containsKey_between_s_106(HashTable sa, HashTable sb, Object k1,
        Object k2)
3359    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
3360                sa..contents = sb..contents & sa..size = sb..size"
3361        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3362        ensures "True" */
3363    {
3364        sa.remove(k1);
3365        /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents))" */
3366        boolean r2a = sa.containsKey(k2);
3367
3368        boolean r2b = sb.containsKey(k2);
3369        sb.remove(k1);
3370
3371        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3372    }
3373
3374    static void remove_containsKey_between_c_106(HashTable sa, HashTable sb, Object k1,
        Object k2)
3375    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
3376                sa..contents = sb..contents & sa..size = sb..size"
3377        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3378        ensures "True" */
3379    {
3380        sa.remove(k1);
3381        /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents)))" */
3382        boolean r2a = sa.containsKey(k2);
3383
3384        boolean r2b = sb.containsKey(k2);
3385        sb.remove(k1);
3386
3387        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3388    }
3389
```

```
    static void remove_containsKey_post_s_107(HashTable sa, HashTable sb, Object k1,
        Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(k1);
        boolean r2a = sa.containsKey(k2);
        /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents))" */

        boolean r2b = sb.containsKey(k2);
        sb.remove(k1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_containsKey_post_c_107(HashTable sa, HashTable sb, Object k1,
        Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(k1);
        boolean r2a = sa.containsKey(k2);
        /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents)))" */

        boolean r2b = sb.containsKey(k2);
        sb.remove(k1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_get_pre_s_108(HashTable sa, HashTable sb, Object k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
        sa.remove(k1);
        Object r2a = sa.get(k2);

        Object r2b = sb.get(k2);
        sb.remove(k1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_get_pre_c_108(HashTable sa, HashTable sb, Object k1, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
        sa.remove(k1);
        Object r2a = sa.get(k2);

        Object r2b = sb.get(k2);
```

```
3449        sb.remove(k1);
3450
3451        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3452    }
3453
3454    static void remove_get_between_s_109(HashTable sa, HashTable sb, Object k1, Object
        k2)
3455    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
3456                sa..contents = sb..contents & sa..size = sb..size"
3457        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3458        ensures "True" */
3459    {
3460        sa.remove(k1);
3461        /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents))" */
3462        Object r2a = sa.get(k2);
3463
3464        Object r2b = sb.get(k2);
3465        sb.remove(k1);
3466
3467        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3468    }
3469
3470    static void remove_get_between_c_109(HashTable sa, HashTable sb, Object k1, Object
        k2)
3471    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
3472                sa..contents = sb..contents & sa..size = sb..size"
3473        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3474        ensures "True" */
3475    {
3476        sa.remove(k1);
3477        /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents)))" */
3478        Object r2a = sa.get(k2);
3479
3480        Object r2b = sb.get(k2);
3481        sb.remove(k1);
3482
3483        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3484    }
3485
3486    static void remove_get_post_s_110(HashTable sa, HashTable sb, Object k1, Object k2)
3487    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
3488                sa..contents = sb..contents & sa..size = sb..size"
3489        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3490        ensures "True" */
3491    {
3492        sa.remove(k1);
3493        Object r2a = sa.get(k2);
3494        /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents))" */
3495
3496        Object r2b = sb.get(k2);
3497        sb.remove(k1);
3498
3499        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3500    }
3501
3502    static void remove_get_post_c_110(HashTable sa, HashTable sb, Object k1, Object k2)
3503    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
3504                sa..contents = sb..contents & sa..size = sb..size"
3505        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3506        ensures "True" */
3507    {
```

```
3508        sa.remove(k1);
3509        Object r2a = sa.get(k2);
3510        /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents)))" */
3511
3512        Object r2b = sb.get(k2);
3513        sb.remove(k1);
3514
3515        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3516    }
3517
3518    static void remove_put_pre_s_111(HashTable sa, HashTable sb, Object k1, Object k2,
        Object v2)
3519    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null & v2 ~= null &
3520                  sa..contents = sb..contents & sa..size = sb..size"
3521        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3522        ensures "True" */
3523    {
3524        /*: assume "k1 ~= k2" */
3525        sa.remove(k1);
3526        Object r2a = sa.put(k2, v2);
3527
3528        Object r2b = sb.put(k2, v2);
3529        sb.remove(k1);
3530
3531        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3532    }
3533
3534    static void remove_put_pre_c_111(HashTable sa, HashTable sb, Object k1, Object k2,
        Object v2)
3535    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null & v2 ~= null &
3536                  sa..contents = sb..contents & sa..size = sb..size"
3537        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3538        ensures "True" */
3539    {
3540        /*: assume "~(k1 ~= k2)" */
3541        sa.remove(k1);
3542        Object r2a = sa.put(k2, v2);
3543
3544        Object r2b = sb.put(k2, v2);
3545        sb.remove(k1);
3546
3547        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3548    }
3549
3550    static void remove_put_between_s_112(HashTable sa, HashTable sb, Object k1, Object
        k2, Object v2)
3551    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null & v2 ~= null &
3552                  sa..contents = sb..contents & sa..size = sb..size"
3553        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3554        ensures "True" */
3555    {
3556        sa.remove(k1);
3557        /*: assume "k1 ~= k2" */
3558        Object r2a = sa.put(k2, v2);
3559
3560        Object r2b = sb.put(k2, v2);
3561        sb.remove(k1);
3562
3563        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3564    }
3565
```

```
3566    static void remove_put_between_c_112(HashTable sa, HashTable sb, Object k1, Object
            k2, Object v2)
3567    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null & v2 ~= null &
3568                    sa..contents = sb..contents & sa..size = sb..size"
3569        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3570        ensures "True" */
3571    {
3572        sa.remove(k1);
3573        /*: assume "~(k1 ~= k2)" */
3574        Object r2a = sa.put(k2, v2);
3575
3576        Object r2b = sb.put(k2, v2);
3577        sb.remove(k1);
3578
3579        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3580    }
3581
3582    static void remove_put_post_s_113(HashTable sa, HashTable sb, Object k1, Object k2,
            Object v2)
3583    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null & v2 ~= null &
3584                    sa..contents = sb..contents & sa..size = sb..size"
3585        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3586        ensures "True" */
3587    {
3588        sa.remove(k1);
3589        Object r2a = sa.put(k2, v2);
3590        /*: assume "k1 ~= k2" */
3591
3592        Object r2b = sb.put(k2, v2);
3593        sb.remove(k1);
3594
3595        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3596    }
3597
3598    static void remove_put_post_c_113(HashTable sa, HashTable sb, Object k1, Object k2,
            Object v2)
3599    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null & v2 ~= null &
3600                    sa..contents = sb..contents & sa..size = sb..size"
3601        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3602        ensures "True" */
3603    {
3604        sa.remove(k1);
3605        Object r2a = sa.put(k2, v2);
3606        /*: assume "~(k1 ~= k2)" */
3607
3608        Object r2b = sb.put(k2, v2);
3609        sb.remove(k1);
3610
3611        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3612    }
3613
3614    static void remove_put_pre_s_114(HashTable sa, HashTable sb, Object k1, Object k2,
            Object v2)
3615    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null & v2 ~= null &
3616                    sa..contents = sb..contents & sa..size = sb..size"
3617        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3618        ensures "True" */
3619    {
3620        /*: assume "k1 ~= k2" */
3621        sa.remove(k1);
3622        sa.put(k2, v2);
```

```
3623
3624        sb.put(k2, v2);
3625        sb.remove(k1);
3626
3627        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
3628    }
3629
3630    static void remove_put_pre_c_114(HashTable sa, HashTable sb, Object k1, Object k2,
        Object v2)
3631    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null & v2 ~= null &
3632                sa..contents = sb..contents & sa..size = sb..size"
3633        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3634        ensures "True" */
3635    {
3636        /*: assume "~(k1 ~= k2)" */
3637        sa.remove(k1);
3638        sa.put(k2, v2);
3639
3640        sb.put(k2, v2);
3641        sb.remove(k1);
3642
3643        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
3644    }
3645
3646    static void remove_put_between_s_115(HashTable sa, HashTable sb, Object k1, Object
        k2, Object v2)
3647    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null & v2 ~= null &
3648                sa..contents = sb..contents & sa..size = sb..size"
3649        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3650        ensures "True" */
3651    {
3652        sa.remove(k1);
3653        /*: assume "k1 ~= k2" */
3654        sa.put(k2, v2);
3655
3656        sb.put(k2, v2);
3657        sb.remove(k1);
3658
3659        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
3660    }
3661
3662    static void remove_put_between_c_115(HashTable sa, HashTable sb, Object k1, Object
        k2, Object v2)
3663    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null & v2 ~= null &
3664                sa..contents = sb..contents & sa..size = sb..size"
3665        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3666        ensures "True" */
3667    {
3668        sa.remove(k1);
3669        /*: assume "~(k1 ~= k2)" */
3670        sa.put(k2, v2);
3671
3672        sb.put(k2, v2);
3673        sb.remove(k1);
3674
3675        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
3676    }
3677
3678    static void remove_put_post_s_116(HashTable sa, HashTable sb, Object k1, Object k2,
        Object v2)
3679    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null & v2 ~= null &
```

```
3680                 sa..contents = sb..contents & sa..size = sb..size"
3681         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3682         ensures "True" */
3683     {
3684         sa.remove(k1);
3685         sa.put(k2, v2);
3686         /*: assume "k1 ~= k2" */
3687 
3688         sb.put(k2, v2);
3689         sb.remove(k1);
3690 
3691         /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
3692     }
3693 
3694     static void remove_put_post_c_116(HashTable sa, HashTable sb, Object k1, Object k2,
        Object v2)
3695     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null & v2 ~= null &
3696                 sa..contents = sb..contents & sa..size = sb..size"
3697         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3698         ensures "True" */
3699     {
3700         sa.remove(k1);
3701         sa.put(k2, v2);
3702         /*: assume "~(k1 ~= k2)" */
3703 
3704         sb.put(k2, v2);
3705         sb.remove(k1);
3706 
3707         /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
3708     }
3709 
3710     static void remove_remove_pre_s_117(HashTable sa, HashTable sb, Object k1, Object
        k2)
3711     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
3712                 sa..contents = sb..contents & sa..size = sb..size"
3713         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3714         ensures "True" */
3715     {
3716         /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
3717         sa.remove(k1);
3718         Object r2a = sa.remove(k2);
3719 
3720         Object r2b = sb.remove(k2);
3721         sb.remove(k1);
3722 
3723         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3724     }
3725 
3726     static void remove_remove_pre_c_117(HashTable sa, HashTable sb, Object k1, Object
        k2)
3727     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
3728                 sa..contents = sb..contents & sa..size = sb..size"
3729         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3730         ensures "True" */
3731     {
3732         /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
3733         sa.remove(k1);
3734         Object r2a = sa.remove(k2);
3735 
3736         Object r2b = sb.remove(k2);
3737         sb.remove(k1);
3738 
```

```
3739        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3740    }
3741
3742    static void remove_remove_between_s_118(HashTable sa, HashTable sb, Object k1,
        Object k2)
3743    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
3744                sa..contents = sb..contents & sa..size = sb..size"
3745        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3746        ensures "True" */
3747    {
3748        sa.remove(k1);
3749        /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents))" */
3750        Object r2a = sa.remove(k2);
3751
3752        Object r2b = sb.remove(k2);
3753        sb.remove(k1);
3754
3755        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3756    }
3757
3758    static void remove_remove_between_c_118(HashTable sa, HashTable sb, Object k1,
        Object k2)
3759    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
3760                sa..contents = sb..contents & sa..size = sb..size"
3761        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3762        ensures "True" */
3763    {
3764        sa.remove(k1);
3765        /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents)))" */
3766        Object r2a = sa.remove(k2);
3767
3768        Object r2b = sb.remove(k2);
3769        sb.remove(k1);
3770
3771        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3772    }
3773
3774    static void remove_remove_post_s_119(HashTable sa, HashTable sb, Object k1, Object
        k2)
3775    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
3776                sa..contents = sb..contents & sa..size = sb..size"
3777        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3778        ensures "True" */
3779    {
3780        sa.remove(k1);
3781        Object r2a = sa.remove(k2);
3782        /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents))" */
3783
3784        Object r2b = sb.remove(k2);
3785        sb.remove(k1);
3786
3787        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3788    }
3789
3790    static void remove_remove_post_c_119(HashTable sa, HashTable sb, Object k1, Object
        k2)
3791    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
3792                sa..contents = sb..contents & sa..size = sb..size"
3793        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3794        ensures "True" */
3795    {
```

```
3796        sa.remove(k1);
3797        Object r2a = sa.remove(k2);
3798        /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents)))" */
3799
3800        Object r2b = sb.remove(k2);
3801        sb.remove(k1);
3802
3803        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3804    }
3805
3806    static void remove_remove_pre_s_120(HashTable sa, HashTable sb, Object k1, Object
        k2)
3807    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
3808                sa..contents = sb..contents & sa..size = sb..size"
3809        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3810        ensures "True" */
3811    {
3812        /*: assume "True" */
3813        sa.remove(k1);
3814        sa.remove(k2);
3815
3816        sb.remove(k2);
3817        sb.remove(k1);
3818
3819        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
3820    }
3821
3822    static void remove_remove_pre_c_120(HashTable sa, HashTable sb, Object k1, Object
        k2)
3823    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
3824                sa..contents = sb..contents & sa..size = sb..size"
3825        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3826        ensures "True" */
3827    {
3828        /*: assume "~(True)" */
3829        sa.remove(k1);
3830        sa.remove(k2);
3831
3832        sb.remove(k2);
3833        sb.remove(k1);
3834
3835        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
3836    }
3837
3838    static void remove_remove_between_s_121(HashTable sa, HashTable sb, Object k1,
        Object k2)
3839    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
3840                sa..contents = sb..contents & sa..size = sb..size"
3841        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3842        ensures "True" */
3843    {
3844        sa.remove(k1);
3845        /*: assume "True" */
3846        sa.remove(k2);
3847
3848        sb.remove(k2);
3849        sb.remove(k1);
3850
3851        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
3852    }
3853
```

```
3854    static void remove_remove_between_c_121(HashTable sa, HashTable sb, Object k1,
            Object k2)
3855    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
3856                  sa..contents = sb..contents & sa..size = sb..size"
3857        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3858        ensures "True" */
3859    {
3860        sa.remove(k1);
3861        /*: assume "~(True)" */
3862        sa.remove(k2);
3863
3864        sb.remove(k2);
3865        sb.remove(k1);
3866
3867        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
3868    }
3869
3870    static void remove_remove_post_s_122(HashTable sa, HashTable sb, Object k1, Object
            k2)
3871    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
3872                  sa..contents = sb..contents & sa..size = sb..size"
3873        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3874        ensures "True" */
3875    {
3876        sa.remove(k1);
3877        sa.remove(k2);
3878        /*: assume "True" */
3879
3880        sb.remove(k2);
3881        sb.remove(k1);
3882
3883        /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
3884    }
3885
3886    static void remove_remove_post_c_122(HashTable sa, HashTable sb, Object k1, Object
            k2)
3887    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            & k2 ~= null &
3888                  sa..contents = sb..contents & sa..size = sb..size"
3889        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3890        ensures "True" */
3891    {
3892        sa.remove(k1);
3893        sa.remove(k2);
3894        /*: assume "~(True)" */
3895
3896        sb.remove(k2);
3897        sb.remove(k1);
3898
3899        /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
3900    }
3901
3902    static void remove_size_pre_s_123(HashTable sa, HashTable sb, Object k1)
3903    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
            &
3904                  sa..contents = sb..contents & sa..size = sb..size"
3905        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3906        ensures "True" */
3907    {
3908        /*: assume "~(EX v. (k1, v) : sa..contents)" */
3909        sa.remove(k1);
3910        int r2a = sa.size();
3911
```

```
        int r2b = sb.size();
        sb.remove(k1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_size_pre_c_123(HashTable sa, HashTable sb, Object k1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(~(EX v. (k1, v) : sa..contents))" */
        sa.remove(k1);
        int r2a = sa.size();

        int r2b = sb.size();
        sb.remove(k1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_size_between_s_124(HashTable sa, HashTable sb, Object k1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(k1);
        /*: assume "~(EX v. (k1, v) : sa..(old contents))" */
        int r2a = sa.size();

        int r2b = sb.size();
        sb.remove(k1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_size_between_c_124(HashTable sa, HashTable sb, Object k1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(k1);
        /*: assume "~(~(EX v. (k1, v) : sa..(old contents)))" */
        int r2a = sa.size();

        int r2b = sb.size();
        sb.remove(k1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void remove_size_post_s_125(HashTable sa, HashTable sb, Object k1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(k1);
```

```
        int r2a = sa.size();
        /*: assume "~(EX v. (k1, v) : sa..(old contents))" */

        int r2b = sb.size();
        sb.remove(k1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void remove_size_post_c_125(HashTable sa, HashTable sb, Object k1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        sa.remove(k1);
        int r2a = sa.size();
        /*: assume "~(~(EX v. (k1, v) : sa..(old contents)))" */

        int r2b = sb.size();
        sb.remove(k1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void size_containsKey_pre_s_126(HashTable sa, HashTable sb, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        /*: assume "True" */
        int r1a = sa.size();
        boolean r2a = sa.containsKey(k2);

        boolean r2b = sb.containsKey(k2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void size_containsKey_pre_c_126(HashTable sa, HashTable sb, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        /*: assume "~(True)" */
        int r1a = sa.size();
        boolean r2a = sa.containsKey(k2);

        boolean r2b = sb.containsKey(k2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void size_containsKey_between_s_127(HashTable sa, HashTable sb, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
```

```
4032    {
4033        int r1a = sa.size();
4034        /*: assume "True" */
4035        boolean r2a = sa.containsKey(k2);
4036
4037        boolean r2b = sb.containsKey(k2);
4038        int r1b = sb.size();
4039
4040        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
4041    }
4042
4043    static void size_containsKey_between_c_127(HashTable sa, HashTable sb, Object k2)
4044    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
            &
4045                    sa..contents = sb..contents & sa..size = sb..size"
4046        ensures "True" */
4047    {
4048        int r1a = sa.size();
4049        /*: assume "~(True)" */
4050        boolean r2a = sa.containsKey(k2);
4051
4052        boolean r2b = sb.containsKey(k2);
4053        int r1b = sb.size();
4054
4055        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
4056    }
4057
4058    static void size_containsKey_post_s_128(HashTable sa, HashTable sb, Object k2)
4059    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
            &
4060                    sa..contents = sb..contents & sa..size = sb..size"
4061        ensures "True" */
4062    {
4063        int r1a = sa.size();
4064        boolean r2a = sa.containsKey(k2);
4065        /*: assume "True" */
4066
4067        boolean r2b = sb.containsKey(k2);
4068        int r1b = sb.size();
4069
4070        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size" */
4071    }
4072
4073    static void size_containsKey_post_c_128(HashTable sa, HashTable sb, Object k2)
4074    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
            &
4075                    sa..contents = sb..contents & sa..size = sb..size"
4076        ensures "True" */
4077    {
4078        int r1a = sa.size();
4079        boolean r2a = sa.containsKey(k2);
4080        /*: assume "~(True)" */
4081
4082        boolean r2b = sb.containsKey(k2);
4083        int r1b = sb.size();
4084
4085        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
4086    }
4087
4088    static void size_get_pre_s_129(HashTable sa, HashTable sb, Object k2)
```

```
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        /*: assume "True" */
        int r1a = sa.size();
        Object r2a = sa.get(k2);

        Object r2b = sb.get(k2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void size_get_pre_c_129(HashTable sa, HashTable sb, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        /*: assume "~(True)" */
        int r1a = sa.size();
        Object r2a = sa.get(k2);

        Object r2b = sb.get(k2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void size_get_between_s_130(HashTable sa, HashTable sb, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "True" */
        Object r2a = sa.get(k2);

        Object r2b = sb.get(k2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void size_get_between_c_130(HashTable sa, HashTable sb, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "~(True)" */
        Object r2a = sa.get(k2);

        Object r2b = sb.get(k2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
```

```
4146     }
4147
4148     static void size_get_post_s_131(HashTable sa, HashTable sb, Object k2)
4149     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
             &
4150                 sa..contents = sb..contents & sa..size = sb..size"
4151         ensures "True" */
4152     {
4153         int r1a = sa.size();
4154         Object r2a = sa.get(k2);
4155         /*: assume "True" */
4156
4157         Object r2b = sb.get(k2);
4158         int r1b = sb.size();
4159
4160         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                 sb..size" */
4161     }
4162
4163     static void size_get_post_c_131(HashTable sa, HashTable sb, Object k2)
4164     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
             &
4165                 sa..contents = sb..contents & sa..size = sb..size"
4166         ensures "True" */
4167     {
4168         int r1a = sa.size();
4169         Object r2a = sa.get(k2);
4170         /*: assume "~(True)" */
4171
4172         Object r2b = sb.get(k2);
4173         int r1b = sb.size();
4174
4175         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                 sb..size)" */
4176     }
4177
4178     static void size_put_pre_s_132(HashTable sa, HashTable sb, Object k2, Object v2)
4179     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
             & v2 ~= null &
4180                 sa..contents = sb..contents & sa..size = sb..size"
4181         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4182         ensures "True" */
4183     {
4184         /*: assume "EX v. (k2, v) : sa..contents" */
4185         int r1a = sa.size();
4186         Object r2a = sa.put(k2, v2);
4187
4188         Object r2b = sb.put(k2, v2);
4189         int r1b = sb.size();
4190
4191         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                 sb..size" */
4192     }
4193
4194     static void size_put_pre_c_132(HashTable sa, HashTable sb, Object k2, Object v2)
4195     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
             & v2 ~= null &
4196                 sa..contents = sb..contents & sa..size = sb..size"
4197         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4198         ensures "True" */
4199     {
4200         /*: assume "~(EX v. (k2, v) : sa..contents)" */
4201         int r1a = sa.size();
4202         Object r2a = sa.put(k2, v2);
4203
```

```
        Object r2b = sb.put(k2, v2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void size_put_between_s_133(HashTable sa, HashTable sb, Object k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "EX v. (k2, v) : sa..contents" */
        Object r2a = sa.put(k2, v2);

        Object r2b = sb.put(k2, v2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void size_put_between_c_133(HashTable sa, HashTable sb, Object k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "~(EX v. (k2, v) : sa..contents)" */
        Object r2a = sa.put(k2, v2);

        Object r2b = sb.put(k2, v2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void size_put_post_s_134(HashTable sa, HashTable sb, Object k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
        Object r2a = sa.put(k2, v2);
        /*: assume "r2a ~= null" */

        Object r2b = sb.put(k2, v2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void size_put_post_c_134(HashTable sa, HashTable sb, Object k2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
        & v2 ~= null &
                  sa..contents = sb..contents & sa..size = sb..size"
```

```
4261         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4262         ensures "True" */
4263     {
4264         int r1a = sa.size();
4265         Object r2a = sa.put(k2, v2);
4266         /*: assume "~(r2a ~= null)" */
4267 
4268         Object r2b = sb.put(k2, v2);
4269         int r1b = sb.size();
4270 
4271         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
                sb..size)" */
4272     }
4273 
4274     static void size_put_pre_s_135(HashTable sa, HashTable sb, Object k2, Object v2)
4275     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
            & v2 ~= null &
4276                 sa..contents = sb..contents & sa..size = sb..size"
4277         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4278         ensures "True" */
4279     {
4280         /*: assume "EX v. (k2, v) : sa..contents" */
4281         int r1a = sa.size();
4282         sa.put(k2, v2);
4283 
4284         sb.put(k2, v2);
4285         int r1b = sb.size();
4286 
4287         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
4288     }
4289 
4290     static void size_put_pre_c_135(HashTable sa, HashTable sb, Object k2, Object v2)
4291     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
            & v2 ~= null &
4292                 sa..contents = sb..contents & sa..size = sb..size"
4293         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4294         ensures "True" */
4295     {
4296         /*: assume "~(EX v. (k2, v) : sa..contents)" */
4297         int r1a = sa.size();
4298         sa.put(k2, v2);
4299 
4300         sb.put(k2, v2);
4301         int r1b = sb.size();
4302 
4303         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
4304     }
4305 
4306     static void size_put_between_s_136(HashTable sa, HashTable sb, Object k2, Object v2)
4307     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
            & v2 ~= null &
4308                 sa..contents = sb..contents & sa..size = sb..size"
4309         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4310         ensures "True" */
4311     {
4312         int r1a = sa.size();
4313         /*: assume "EX v. (k2, v) : sa..contents" */
4314         sa.put(k2, v2);
4315 
4316         sb.put(k2, v2);
4317         int r1b = sb.size();
4318 
4319         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
4320     }
4321 
```

```
4322    static void size_put_between_c_136(HashTable sa, HashTable sb, Object k2, Object v2)
4323    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
            & v2 ~= null &
4324                sa..contents = sb..contents & sa..size = sb..size"
4325        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4326        ensures "True" */
4327    {
4328        int r1a = sa.size();
4329        /*: assume "~(EX v. (k2, v) : sa..contents)" */
4330        sa.put(k2, v2);
4331
4332        sb.put(k2, v2);
4333        int r1b = sb.size();
4334
4335        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
4336    }
4337
4338    static void size_put_post_s_137(HashTable sa, HashTable sb, Object k2, Object v2)
4339    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
            & v2 ~= null &
4340                sa..contents = sb..contents & sa..size = sb..size"
4341        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4342        ensures "True" */
4343    {
4344        int r1a = sa.size();
4345        /*: ghost specvar sa__contents :: "(obj * obj) set" = "sa..contents" */
4346        sa.put(k2, v2);
4347        /*: assume "EX v. (k2, v) : sa__contents" */
4348
4349        sb.put(k2, v2);
4350        int r1b = sb.size();
4351
4352        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
4353    }
4354
4355    static void size_put_post_c_137(HashTable sa, HashTable sb, Object k2, Object v2)
4356    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
            & v2 ~= null &
4357                sa..contents = sb..contents & sa..size = sb..size"
4358        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4359        ensures "True" */
4360    {
4361        int r1a = sa.size();
4362        /*: ghost specvar sa__contents :: "(obj * obj) set" = "sa..contents" */
4363        sa.put(k2, v2);
4364        /*: assume "~(EX v. (k2, v) : sa__contents)" */
4365
4366        sb.put(k2, v2);
4367        int r1b = sb.size();
4368
4369        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
4370    }
4371
4372    static void size_remove_pre_s_138(HashTable sa, HashTable sb, Object k2)
4373    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
            &
4374                sa..contents = sb..contents & sa..size = sb..size"
4375        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4376        ensures "True" */
4377    {
4378        /*: assume "~(EX v. (k2, v) : sa..contents)" */
4379        int r1a = sa.size();
4380        Object r2a = sa.remove(k2);
4381
4382        Object r2b = sb.remove(k2);
```

```
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void size_remove_pre_c_138(HashTable sa, HashTable sb, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(~(EX v. (k2, v) : sa..contents))" */
        int r1a = sa.size();
        Object r2a = sa.remove(k2);

        Object r2b = sb.remove(k2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void size_remove_between_s_139(HashTable sa, HashTable sb, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "~(EX v. (k2, v) : sa..contents)" */
        Object r2a = sa.remove(k2);

        Object r2b = sb.remove(k2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void size_remove_between_c_139(HashTable sa, HashTable sb, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "~(~(EX v. (k2, v) : sa..contents))" */
        Object r2a = sa.remove(k2);

        Object r2b = sb.remove(k2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void size_remove_post_s_140(HashTable sa, HashTable sb, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
```

```
        ensures "True" */
    {
        int r1a = sa.size();
        Object r2a = sa.remove(k2);
        /*: assume "r2a = null" */

        Object r2b = sb.remove(k2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void size_remove_post_c_140(HashTable sa, HashTable sb, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
        Object r2a = sa.remove(k2);
        /*: assume "~(r2a = null)" */

        Object r2b = sb.remove(k2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void size_remove_pre_s_141(HashTable sa, HashTable sb, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(EX v. (k2, v) : sa..contents)" */
        int r1a = sa.size();
        sa.remove(k2);

        sb.remove(k2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
    }

    static void size_remove_pre_c_141(HashTable sa, HashTable sb, Object k2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
        &
                   sa..contents = sb..contents & sa..size = sb..size"
        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
        ensures "True" */
    {
        /*: assume "~(~(EX v. (k2, v) : sa..contents))" */
        int r1a = sa.size();
        sa.remove(k2);

        sb.remove(k2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }
```

```
4500    static void size_remove_between_s_142(HashTable sa, HashTable sb, Object k2)
4501    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
            &
4502                sa..contents = sb..contents & sa..size = sb..size"
4503        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4504        ensures "True" */
4505    {
4506        int r1a = sa.size();
4507        /*: assume "~(EX v. (k2, v) : sa..contents)" */
4508        sa.remove(k2);
4509
4510        sb.remove(k2);
4511        int r1b = sb.size();
4512
4513        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
4514    }
4515
4516    static void size_remove_between_c_142(HashTable sa, HashTable sb, Object k2)
4517    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
            &
4518                sa..contents = sb..contents & sa..size = sb..size"
4519        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4520        ensures "True" */
4521    {
4522        int r1a = sa.size();
4523        /*: assume "~(~(EX v. (k2, v) : sa..contents))" */
4524        sa.remove(k2);
4525
4526        sb.remove(k2);
4527        int r1b = sb.size();
4528
4529        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
4530    }
4531
4532    static void size_remove_post_s_143(HashTable sa, HashTable sb, Object k2)
4533    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
            &
4534                sa..contents = sb..contents & sa..size = sb..size"
4535        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4536        ensures "True" */
4537    {
4538        int r1a = sa.size();
4539        /*: ghost specvar sa__contents :: "(obj * obj) set" = "sa..contents" */
4540        sa.remove(k2);
4541        /*: assume "~(EX v. (k2, v) : sa__contents)" */
4542
4543        sb.remove(k2);
4544        int r1b = sb.size();
4545
4546        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
4547    }
4548
4549    static void size_remove_post_c_143(HashTable sa, HashTable sb, Object k2)
4550    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
            &
4551                sa..contents = sb..contents & sa..size = sb..size"
4552        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4553        ensures "True" */
4554    {
4555        int r1a = sa.size();
4556        /*: ghost specvar sa__contents :: "(obj * obj) set" = "sa..contents" */
4557        sa.remove(k2);
4558        /*: assume "~(~(EX v. (k2, v) : sa__contents))" */
4559
4560        sb.remove(k2);
```

```
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
    }

    static void size_size_pre_s_144(HashTable sa, HashTable sb)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        /*: assume "True" */
        int r1a = sa.size();
        int r2a = sa.size();

        int r2b = sb.size();
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void size_size_pre_c_144(HashTable sa, HashTable sb)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        /*: assume "~(True)" */
        int r1a = sa.size();
        int r2a = sa.size();

        int r2b = sb.size();
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void size_size_between_s_145(HashTable sa, HashTable sb)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "True" */
        int r2a = sa.size();

        int r2b = sb.size();
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void size_size_between_c_145(HashTable sa, HashTable sb)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "~(True)" */
        int r2a = sa.size();

        int r2b = sb.size();
        int r1b = sb.size();

```

```
        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

    static void size_size_post_s_146(HashTable sa, HashTable sb)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                   sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
        int r2a = sa.size();
        /*: assume "True" */

        int r2b = sb.size();
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size" */
    }

    static void size_size_post_c_146(HashTable sa, HashTable sb)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                   sa..contents = sb..contents & sa..size = sb..size"
        ensures "True" */
    {
        int r1a = sa.size();
        int r2a = sa.size();
        /*: assume "~(True)" */

        int r2b = sb.size();
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
            sb..size)" */
    }

}
```

B.5.3 Inverse Testing Methods

Listing 15. HashTableInv.java

```
class HashTableInv {
    static void put_0(HashTable s, Object k, Object v)
    /*: requires "s ~= null & s..init & k ~= null & v ~= null"
        modifies "s..contents", "s..size"
        ensures "True" */
    {
        Object r = s.put(k, v);
        if (r != null) { s.put(k, r); } else { s.remove(k); }

        /*: assert "s..contents = s..(old contents) & s..size = s..(old size)" */
    }

    static void remove_1(HashTable s, Object k)
    /*: requires "s ~= null & s..init & k ~= null"
        modifies "s..contents", "s..size"
        ensures "True" */
    {
        Object r = s.remove(k);
        if (r != null) { s.put(k, r); }

        /*: assert "s..contents = s..(old contents) & s..size = s..(old size)" */
    }

```

```
}
```

B.6 ArrayList

B.6.1 Data Structure

Listing 16. ArrayList.java

```
public class ArrayList
{
    private Object[] elementData;
    private int size;

    /*: public specvar init :: bool;
        vardefs "init == elementData ~= null"

        invariant InitInv: "~init --> size = 0 & elementData = null"
        invariant ElementDataInv: "init --> elementData : hidden"
        invariant NotHiddenInv: "init --> (ALL i. ((0 <= i & i < size) -->
            elementData.[i] ~: hidden))"
        invariant InjInv: "ALL x y. x ~= y & x..elementData ~= null --> x..elementData
            ~= y..elementData";
        invariant SizeInv: "init --> 0 <= size & size <= elementData..Array.length"

        public specvar contents :: "(int * obj) set";
        vardefs "contents == {(i, n). 0 <= i & i < size & n = elementData.[i]}"
        public specvar csize :: int;
        vardefs "csize == size"
        public specvar msize :: int;
        vardefs "msize == elementData..Array.length"

        public ensured invariant "init --> (ALL i v. (i, v) : contents --> 0 <= i & i <
            csize)"
        public ensured invariant RangeInvInj: "init --> (ALL i v1 v2. (i, v1) :
            contents & (i, v2) : contents --> v1 = v2)" */

    public ArrayList(int initialCapacity)
    /*: requires "~init & 0 < initialCapacity"
        modifies "init", "msize"
        ensures "init & contents = {} & csize = 0 & msize = initialCapacity" */
    {
        elementData = new /*: hidden */ Object[initialCapacity];
    }

    private  void _ensureCapacity(int minCapacity)
    /*: requires "init & theinvs"
        modifies "Array.arrayState", "elementData", "msize", "ArrayList.hidden"
        ensures "minCapacity <= msize & old msize <= msize & (ALL x. x..init =
            (fieldRead (old ArrayList.init) x)) & (ALL a j. a ~= elementData --> a.[j] =
            arrayRead (old arrayState) a j) & (ALL j. ((0 <= j & j < size) -->
            elementData.[j] = arrayRead (old arrayState) (old elementData) j)) & (ALL
            o1. (o1 : old alloc) --> ((o1 : old hidden) = (o1 : hidden))) & theinvs" */
    {
        int oldCapacity = elementData.length;
        if (minCapacity > oldCapacity) {
            Object oldData[] = elementData;
            int newCapacity = (oldCapacity * 3)/2 + 1;
            if (newCapacity < minCapacity)
                newCapacity = minCapacity;
            elementData = new /*: hidden */ Object[newCapacity];
            int i = 0;
            while /*: invariant "0 <= i & (ALL j. ((0 <= j & j < size --> oldData.[j] =
                arrayRead (old arrayState) (old elementData) j))) & (ALL j. ((0 <= j & j
                < i) --> elementData.[j] = oldData.[j])) & (ALL a j. a ~= elementData
                --> a.[j] = arrayRead (old arrayState) a j)" */ (i < size) {
                elementData[i] = oldData[i];
```

```
                i = i + 1;
            }
        }
    }

    public int size()
    /*: requires "init"
        ensures "result = csize" */
    {
        return size;
    }

    public int indexOf(Object elem)
    /*: requires "init"
        ensures "(~(EX i. (i, elem) : contents & 0 <= i & i < csize) --> result = -1) &
            ((EX i. (i, elem) : contents & 0 <= i & i < csize) --> (result, elem) :
            contents & ~(EX i. (i, elem) : contents & 0 <= i & i < result) & 0 <= result
            & result < csize)" */
    {
        return indexOfInt(elem);
    }

    private int indexOfInt(Object elem)
    /*: requires "init & theinvs"
        ensures "(~(EX i. (i, elem) : contents & 0 <= i & i < csize) --> result = -1) &
            ((EX i. (i, elem) : contents & 0 <= i & i < csize) --> (result, elem) :
            contents & ~(EX i. (i, elem) : contents & 0 <= i & i < result) & 0 <= result
            & result < csize) & theinvs" */
    {
        int i = 0;

        while /*: invariant "0 <= i & (ALL j. 0 <= j & j < i --> elem ~=
            elementData.[j])" */ (i < size) {
            if (elementData[i] == elem)
                return i;
            i = i + 1;
        }
        return -1;
    }

    public int lastIndexOf(Object elem)
    /*: requires "init"
        ensures "(~(EX i. (i, elem) : contents & 0 <= i & i < csize) --> result = -1) &
            ((EX i. (i, elem) : contents & 0 <= i & i < csize) --> (result, elem) :
            contents & ~(EX i. (i, elem) : contents & result < i & i < csize) & 0 <=
            result & result < csize)" */

    {
        int i = size - 1;
        while /*: invariant "i < size & (ALL j. i < j & j < size --> elem ~=
            elementData.[j])" */ (i >= 0) {
            if (elementData[i] == elem)
                return i;
            i = i - 1;
        }
        return -1;
    }

    public Object get(int index)
    /*: requires "init & 0 <= index & index < csize"
        ensures "(index, result) : contents & (ALL i v1 v2. (i, v1) : contents & (i,
            v2) : contents --> v1 = v2)" */
    {
        return elementData[index];
    }
```

```
101
102     public Object set(int index, Object element)
103     /*: requires "init & 0 <= index & index < csize"
104         modifies "contents"
105         ensures "(index, result) : old contents & contents = (old contents - {(index,
                result)}) Un {(index, element)}" */
106     {
107         Object oldValue = elementData[index];
108         elementData[index] = element;
109         return oldValue;
110     }
111
112     public void add_at(int index, Object element)
113     /*: requires "comment ''addAtPre'' (init & 0 <= index & index <= csize)"
114         modifies "contents", "csize", "msize"
115         ensures "(ALL j e.  (0 <= j & j < index --> ((j, e) : contents) = ((j, e) : old
                contents))) & ((index, element) : contents) & (ALL j e. (index < j & j <
                csize --> ((j, e) : contents) = ((j - 1, e) : old contents))) & (csize =
                (old csize) + 1) & (msize >= (old msize)) & (csize <= msize) & (ALL i v. (i,
                v) : contents --> 0 <= i & i < csize) & (ALL i v1 v2. (i, v1) : contents &
                (i, v2) : contents --> v1 = v2)" */
116     {
117         _ensureCapacity(size + 1);
118         int i = size - 1;
119         while /*: invariant "index - 1 <= i & i <= size - 1 & (ALL a j. ((a ~: hidden)
                --> a.[j] = arrayRead (old Array.arrayState) a j)) & ((i = size - 1) -->
                (ALL j. ((0 <= j & j < size) --> elementData.[j] ~: hidden))) & ((i < size -
                1) --> (ALL j. ((0 <= j & j < size + 1) --> elementData.[j] ~: hidden))) &
                ((ALL j. ((0 <= j & j <= i) --> elementData.[j] = arrayRead (old
                Array.arrayState) (old elementData) j))) & ((ALL j. ((0 <= j & j <= index &
                j < size) --> elementData.[j] = arrayRead (old Array.arrayState) (old
                elementData) j))) & ((ALL j. ((i < j & j < size) --> elementData.[j + 1] =
                arrayRead (old Array.arrayState) (old elementData) j))) & (ALL x i. (x ~=
                elementData & x ~= (old elementData)) --> (x.[i] = (arrayRead (old
                Array.arrayState) (old x) i)))" */ (index <= i) {
120             elementData[i + 1] = elementData[i];
121             i = i - 1;
122         }
123         elementData[index] = element;
124         size = size + 1;
125         /*: note InvMeans: "(ALL j. ((0 <= j & j < index & j < (old size)) -->
                elementData.[j] = arrayRead (old Array.arrayState) (old elementData) j)) &
                (ALL j. ((index <= j & j < (old size)) --> elementData.[j + 1] = arrayRead
                (old Array.arrayState) (old elementData) j))"; */
126         /*: note IndexRange: "0 <= index & index <= old size"; */
127         /*: note CsizeIsSize: "csize = size"; */
128         /*: note PostCond: "(ALL j e.  (0 <= j & j < index --> ((j, e) : contents) =
                ((j, e) : old contents)) & (index < j & j < csize --> ((j, e) : contents) =
                ((j - 1, e) : old contents)))" from InvMeans, PostCond, contents_def,
                IndexRange, CsizeIsSize; */
129         /*: note OtherPostCond: "(index, element) : contents" from OtherPostCond,
                contents_def, IndexRange; */
130     }
131
132     public Object remove_at(int index)
133     /*: requires "init & 0 <= index & index < csize"
134         modifies "contents", "csize"
135         ensures "((index, result) : old contents) & (csize = old csize - 1) & (ALL j e.
                 ((0 <= j & j < index) --> ((j, e) : contents) = ((j, e) : old contents)) &
                ((index <= j & j < csize) --> ((j, e) : contents) = ((j + 1, e) : old
                contents))) & (ALL i v. (i, v) : contents --> 0 <= i & i < csize)" */
136     {
137         Object oldValue = elementData[index];
138         int i = index;
```

```
        while /*: invariant "index <= i & (ALL j. ((0 <= j & j < index) -->
            elementData.[j] = (arrayRead (old Array.arrayState) (old elementData) j))) &
            (ALL j. ((index <= j & j < i) --> elementData.[j] = (arrayRead (old
            Array.arrayState) elementData (j + 1)))) & (ALL j. ((i <= j & j < size) -->
            elementData.[j] = (arrayRead (old Array.arrayState) (old elementData) j))) &
            (ALL x i. x ~= elementData --> x.[i] = arrayRead (old Array.arrayState) (old
            x) i)" */ (i < size - 1) {
            elementData[i] = elementData[i+1];
            i = i + 1;
        }
        size = size - 1;
        elementData[size] = null;

        return oldValue;
    }
}
```

B.6.2 Commutativity Testing Methods

Listing 17 presents all 486 automatically generated commutativity testing methods. The 429 methods of them that verify as generated are shown in Listing 18 separately. The remaining 57 methods require additional proof language commands to verify, and are presented in Listing 19 with the added proof language commands.

Listing 17. `ArrayListComm.java`

```
class ArrayListComm {
    static void add_at_add_at_pre_s_0(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "(i1 < i2 & i2 <= sa..csize & (i2 - 1, v2) : sa..contents & 0 <= i2
            - 1 & i2 - 1 < sa..csize) | (i1 = i2 & v1 = v2) | (i1 > i2 & (i1 - 1, v1) :
            sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize)" */
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        sb.add_at(i2, v2);
        sb.add_at(i1, v1);

        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void add_at_add_at_pre_c_0(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "~((i1 < i2 & i2 <= sa..csize & (i2 - 1, v2) : sa..contents & 0 <=
            i2 - 1 & i2 - 1 < sa..csize) | (i1 = i2 & v1 = v2) | (i1 > i2 & (i1 - 1, v1)
            : sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize))" */
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        /*: assume "0 <= i2 & i2 <= sb..csize" */
        sb.add_at(i2, v2);
```

```
        /*: assume "0 <= i1 & i1 <= sb..csize" */
        sb.add_at(i1, v1);

        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
    }

    static void add_at_add_at_between_s_1(ArrayList sa, ArrayList sb, int i1, Object
        v1, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "(i1 < i2 & i2 <= sa..csize - 1 & (i2, v2) : sa..contents & 0 <= i2
            & i2 < sa..csize) | (i1 = i2 & v1 = v2) | (i1 > i2 & (i1 - 1, v1) :
            sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize)" */
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        sb.add_at(i2, v2);
        sb.add_at(i1, v1);

        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void add_at_add_at_between_c_1(ArrayList sa, ArrayList sb, int i1, Object
        v1, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "~((i1 < i2 & i2 <= sa..csize - 1 & (i2, v2) : sa..contents & 0 <=
            i2 & i2 < sa..csize) | (i1 = i2 & v1 = v2) | (i1 > i2 & (i1 - 1, v1) :
            sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize))" */
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        /*: assume "0 <= i2 & i2 <= sb..csize" */
        sb.add_at(i2, v2);
        /*: assume "0 <= i1 & i1 <= sb..csize" */
        sb.add_at(i1, v1);

        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
    }

    static void add_at_add_at_post_s_2(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);
```

```
88          /*: assume "(i1 < i2 & i2 <= sa..csize - 2 & (i2 + 1, v2) : sa..contents & 0 <=
                i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & v1 = v2) | (i1 > i2 & (i1, v1) :
                sa..contents & 0 <= i1 & i1 < sa..csize)" */
89
90          sb.add_at(i2, v2);
91          sb.add_at(i1, v1);
92
93          /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
94      }
95
96      static void add_at_add_at_post_c_2(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2, Object v2)
97      /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
98                    sa..contents = sb..contents & sa..csize = sb..csize"
99          modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
100         ensures "True" */
101     {
102         /*: assume "0 <= i1 & i1 <= sa..csize" */
103         sa.add_at(i1, v1);
104         /*: assume "0 <= i2 & i2 <= sa..csize" */
105         sa.add_at(i2, v2);
106         /*: assume "~((i1 < i2 & i2 <= sa..csize - 2 & (i2 + 1, v2) : sa..contents & 0
                <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & v1 = v2) | (i1 > i2 & (i1, v1)
                : sa..contents & 0 <= i1 & i1 < sa..csize))" */
107
108         /*: assume "0 <= i2 & i2 <= sb..csize" */
109         sb.add_at(i2, v2);
110         /*: assume "0 <= i1 & i1 <= sb..csize" */
111         sb.add_at(i1, v1);
112
113         /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
114     }
115
116     static void add_at_get_pre_s_3(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2)
117     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
118                   sa..contents = sb..contents & sa..csize = sb..csize"
119         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
120         ensures "True" */
121     {
122         /*: assume "(i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents) =
                ((i2, v) : sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2
                < sa..csize) | (i1 = i2 & i2 < sa..csize & (i1, v1) : sa..contents & 0 <= i1
                & i1 < sa..csize) | i1 > i2" */
123         /*: assume "0 <= i1 & i1 <= sa..csize" */
124         sa.add_at(i1, v1);
125         /*: assume "0 <= i2 & i2 < sa..csize" */
126         Object r2a = sa.get(i2);
127
128         Object r2b = sb.get(i2);
129         sb.add_at(i1, v1);
130
131         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
132     }
133
134     static void add_at_get_pre_c_3(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2)
135     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
136                   sa..contents = sb..contents & sa..csize = sb..csize"
137         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
138         ensures "True" */
139     {
```

```
        /*: assume "~((i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents)
            = ((i2, v) : sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 &
            i2 < sa..csize) | (i1 = i2 & i2 < sa..csize & (i1, v1) : sa..contents & 0 <=
            i1 & i1 < sa..csize) | i1 > i2)" */
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.get(i2);
        /*: assume "0 <= i1 & i1 <= sb..csize" */
        sb.add_at(i1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void add_at_get_between_s_4(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "(i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa..contents) =
            ((i2 + 1, v) : sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2
            + 1 < sa..csize) | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, v1) :
            sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | i1 > i2" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        Object r2b = sb.get(i2);
        sb.add_at(i1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void add_at_get_between_c_4(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "~((i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa..contents)
            = ((i2 + 1, v) : sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 &
            i2 + 1 < sa..csize) | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, v1) :
            sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | i1 > i2)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.get(i2);
        /*: assume "0 <= i1 & i1 <= sb..csize" */
        sb.add_at(i1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
```

```
190     }
191
192     static void add_at_get_post_s_5(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2)
193     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
194                     sa..contents = sb..contents & sa..csize = sb..csize"
195         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
196         ensures "True" */
197     {
198         /*: assume "0 <= i1 & i1 <= sa..csize" */
199         sa.add_at(i1, v1);
200         /*: assume "0 <= i2 & i2 < sa..csize" */
201         Object r2a = sa.get(i2);
202         /*: assume "(i1 < i2 & i2 < sa..csize - 1 & (i2 + 1, r2a) : sa..contents & 0 <=
                i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, v1)
                : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | i1 > i2" */
203
204         Object r2b = sb.get(i2);
205         sb.add_at(i1, v1);
206
207         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
208     }
209
210     static void add_at_get_post_c_5(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2)
211     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
212                     sa..contents = sb..contents & sa..csize = sb..csize"
213         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
214         ensures "True" */
215     {
216         /*: assume "0 <= i1 & i1 <= sa..csize" */
217         sa.add_at(i1, v1);
218         /*: assume "0 <= i2 & i2 < sa..csize" */
219         Object r2a = sa.get(i2);
220         /*: assume "~((i1 < i2 & i2 < sa..csize - 1 & (i2 + 1, r2a) : sa..contents & 0
                <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1,
                v1) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | i1 > i2)" */
221
222         /*: assume "0 <= i2 & i2 < sb..csize" */
223         Object r2b = sb.get(i2);
224         /*: assume "0 <= i1 & i1 <= sb..csize" */
225         sb.add_at(i1, v1);
226
227         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
228     }
229
230     static void add_at_indexOf_pre_s_6(ArrayList sa, ArrayList sb, int i1, Object v1,
            Object v2)
231     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
232                     sa..contents = sb..contents & sa..csize = sb..csize"
233         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
234         ensures "True" */
235     {
236         /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
                v2) | (EX i. (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) :
                sa..contents & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 <
                sa..csize & v1 = v2)" */
237         /*: assume "0 <= i1 & i1 <= sa..csize" */
238         sa.add_at(i1, v1);
239         int r2a = sa.indexOf(v2);
240
```

```
241         int r2b = sb.indexOf(v2);
242         sb.add_at(i1, v1);
243 
244         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
245     }
246 
247     static void add_at_indexOf_pre_c_6(ArrayList sa, ArrayList sb, int i1, Object v1,
        Object v2)
248     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
249                 sa..contents = sb..contents & sa..csize = sb..csize"
250         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
251         ensures "True" */
252     {
253         /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
                v2) | (EX i. (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) :
                sa..contents & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 <
                sa..csize & v1 = v2))" */
254         /*: assume "0 <= i1 & i1 <= sa..csize" */
255         sa.add_at(i1, v1);
256         int r2a = sa.indexOf(v2);
257 
258         int r2b = sb.indexOf(v2);
259         /*: assume "0 <= i1 & i1 <= sb..csize" */
260         sb.add_at(i1, v1);
261 
262         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
263     }
264 
265     static void add_at_indexOf_between_s_7(ArrayList sa, ArrayList sb, int i1, Object
        v1, Object v2)
266     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
267                 sa..contents = sb..contents & sa..csize = sb..csize"
268         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
269         ensures "True" */
270     {
271         /*: assume "0 <= i1 & i1 <= sa..csize" */
272         sa.add_at(i1, v1);
273         /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
                (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) : sa..contents
                & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize &
                (i1 + 1, v2) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
274         int r2a = sa.indexOf(v2);
275 
276         int r2b = sb.indexOf(v2);
277         sb.add_at(i1, v1);
278 
279         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
280     }
281 
282     static void add_at_indexOf_between_c_7(ArrayList sa, ArrayList sb, int i1, Object
        v1, Object v2)
283     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
284                 sa..contents = sb..contents & sa..csize = sb..csize"
285         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
286         ensures "True" */
287     {
288         /*: assume "0 <= i1 & i1 <= sa..csize" */
289         sa.add_at(i1, v1);
290         /*: assume "~(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
                (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) : sa..contents
```

```
            & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize &
            (i1 + 1, v2) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
        int r2a = sa.indexOf(v2);

        int r2b = sb.indexOf(v2);
        /*: assume "0 <= i1 & i1 <= sb..csize" */
        sb.add_at(i1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void add_at_indexOf_post_s_8(ArrayList sa, ArrayList sb, int i1, Object v1,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        int r2a = sa.indexOf(v2);
        /*: assume "r2a < 0 | (0 <= r2a & r2a < i1) | (r2a = i1 & (i1 + 1, v2) :
            sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */

        int r2b = sb.indexOf(v2);
        sb.add_at(i1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void add_at_indexOf_post_c_8(ArrayList sa, ArrayList sb, int i1, Object v1,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        int r2a = sa.indexOf(v2);
        /*: assume "~(r2a < 0 | (0 <= r2a & r2a < i1) | (r2a = i1 & (i1 + 1, v2) :
            sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */

        int r2b = sb.indexOf(v2);
        /*: assume "0 <= i1 & i1 <= sb..csize" */
        sb.add_at(i1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void add_at_lastIndexOf_pre_s_9(ArrayList sa, ArrayList sb, int i1, Object
        v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
            v2) | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
            sa..contents & i1 <= i & i < sa..csize) & v1 ~= v2)" */
```

```
342         /*: assume "0 <= i1 & i1 <= sa..csize" */
343         sa.add_at(i1, v1);
344         int r2a = sa.lastIndexOf(v2);
345 
346         int r2b = sb.lastIndexOf(v2);
347         sb.add_at(i1, v1);
348 
349         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
350     }
351 
352     static void add_at_lastIndexOf_pre_c_9(ArrayList sa, ArrayList sb, int i1, Object
            v1, Object v2)
353     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
354                     sa..contents = sb..contents & sa..csize = sb..csize"
355         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
356         ensures "True" */
357     {
358         /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
                v2) | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
                sa..contents & i1 <= i & i < sa..csize) & v1 ~= v2))" */
359         /*: assume "0 <= i1 & i1 <= sa..csize" */
360         sa.add_at(i1, v1);
361         int r2a = sa.lastIndexOf(v2);
362 
363         int r2b = sb.lastIndexOf(v2);
364         /*: assume "0 <= i1 & i1 <= sb..csize" */
365         sb.add_at(i1, v1);
366 
367         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
368     }
369 
370     static void add_at_lastIndexOf_between_s_10(ArrayList sa, ArrayList sb, int i1,
            Object v1, Object v2)
371     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
372                     sa..contents = sb..contents & sa..csize = sb..csize"
373         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
374         ensures "True" */
375     {
376         /*: assume "0 <= i1 & i1 <= sa..csize" */
377         sa.add_at(i1, v1);
378         /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | ((EX i.
                (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) : sa..contents &
                i1 <= i & i < sa..csize))" */
379         int r2a = sa.lastIndexOf(v2);
380 
381         int r2b = sb.lastIndexOf(v2);
382         sb.add_at(i1, v1);
383 
384         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
385     }
386 
387     static void add_at_lastIndexOf_between_c_10(ArrayList sa, ArrayList sb, int i1,
            Object v1, Object v2)
388     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
389                     sa..contents = sb..contents & sa..csize = sb..csize"
390         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
391         ensures "True" */
392     {
393         /*: assume "0 <= i1 & i1 <= sa..csize" */
394         sa.add_at(i1, v1);
```

```
        /*: assume "~(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | ((EX
            i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
            sa..contents & i1 <= i & i < sa..csize)))" */
        int r2a = sa.lastIndexOf(v2);

        int r2b = sb.lastIndexOf(v2);
        /*: assume "0 <= i1 & i1 <= sb..csize" */
        sb.add_at(i1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void add_at_lastIndexOf_post_s_11(ArrayList sa, ArrayList sb, int i1, Object
        v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        int r2a = sa.lastIndexOf(v2);
        /*: assume "r2a < 0 | (0 <= r2a & r2a < i1)" */

        int r2b = sb.lastIndexOf(v2);
        sb.add_at(i1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void add_at_lastIndexOf_post_c_11(ArrayList sa, ArrayList sb, int i1, Object
        v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        int r2a = sa.lastIndexOf(v2);
        /*: assume "~(r2a < 0 | (0 <= r2a & r2a < i1))" */

        int r2b = sb.lastIndexOf(v2);
        /*: assume "0 <= i1 & i1 <= sb..csize" */
        sb.add_at(i1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void add_at_remove_at_pre_s_12(ArrayList sa, ArrayList sb, int i1, Object
        v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "(i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents) =
            ((i2, v) : sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2
            < sa..csize) | (sa..csize - 1 >= i1 & i1 = i2 & (i1, v1) : sa..contents & 0
```

```
        <= i1 & i1 < sa..csize) | (sa..csize - 1 >= i1 & i1 > i2 & (i1, v1) :
        sa..contents & 0 <= i1 & i1 < sa..csize)" */
    /*: assume "0 <= i1 & i1 <= sa..csize" */
    sa.add_at(i1, v1);
    /*: assume "0 <= i2 & i2 < sa..csize" */
    Object r2a = sa.remove_at(i2);

    Object r2b = sb.remove_at(i2);
    sb.add_at(i1, v1);

    /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
}

static void add_at_remove_at_pre_c_12(ArrayList sa, ArrayList sb, int i1, Object
    v1, int i2)
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
              sa..contents = sb..contents & sa..csize = sb..csize"
    modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
        "sb..msize"
    ensures "True" */
{
    /*: assume "~((i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents)
        = ((i2, v) : sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 &
        i2 < sa..csize) | (sa..csize - 1 >= i1 & i1 = i2 & (i1, v1) : sa..contents &
        0 <= i1 & i1 < sa..csize) | (sa..csize - 1 >= i1 & i1 > i2 & (i1, v1) :
        sa..contents & 0 <= i1 & i1 < sa..csize))" */
    /*: assume "0 <= i1 & i1 <= sa..csize" */
    sa.add_at(i1, v1);
    /*: assume "0 <= i2 & i2 < sa..csize" */
    Object r2a = sa.remove_at(i2);

    /*: assume "0 <= i2 & i2 < sb..csize" */
    Object r2b = sb.remove_at(i2);
    /*: assume "0 <= i1 & i1 <= sb..csize" */
    sb.add_at(i1, v1);

    /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
}

static void add_at_remove_at_between_s_13(ArrayList sa, ArrayList sb, int i1,
    Object v1, int i2)
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
              sa..contents = sb..contents & sa..csize = sb..csize"
    modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
        "sb..msize"
    ensures "True" */
{
    /*: assume "0 <= i1 & i1 <= sa..csize" */
    sa.add_at(i1, v1);
    /*: assume "(i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa..contents) =
        ((i2 + 1, v) : sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2
        + 1 < sa..csize) | (sa..csize - 2 >= i1 & i1 = i2 & (i1 + 1, v1) :
        sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 2 >= i1 & i1
        > i2 & (i1 + 1, v1) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
    /*: assume "0 <= i2 & i2 < sa..csize" */
    Object r2a = sa.remove_at(i2);

    Object r2b = sb.remove_at(i2);
    sb.add_at(i1, v1);

    /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
}

```

```
    static void add_at_remove_at_between_c_13(ArrayList sa, ArrayList sb, int i1,
        Object v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "~((i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa..contents)
            = ((i2 + 1, v) : sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 &
            i2 + 1 < sa..csize) | (sa..csize - 2 >= i1 & i1 = i2 & (i1 + 1, v1) :
            sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 2 >= i1 & i1
            > i2 & (i1 + 1, v1) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 <= sb..csize" */
        sb.add_at(i1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void add_at_remove_at_post_s_14(ArrayList sa, ArrayList sb, int i1, Object
        v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);
        /*: assume "(i1 < i2 & i2 < sa..csize & (i2, r2a) : sa..contents & 0 <= i2 & i2
            < sa..csize) | (sa..csize - 1 >= i1 & i1 = i2 & (i1, v1) : sa..contents & 0
            <= i1 & i1 < sa..csize) | (sa..csize - 1 >= i1 & i1 > i2 & (i1, v1) :
            sa..contents & 0 <= i1 & i1 < sa..csize)" */

        Object r2b = sb.remove_at(i2);
        sb.add_at(i1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void add_at_remove_at_post_c_14(ArrayList sa, ArrayList sb, int i1, Object
        v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);
        /*: assume "~((i1 < i2 & i2 < sa..csize & (i2, r2a) : sa..contents & 0 <= i2 &
            i2 < sa..csize) | (sa..csize - 1 >= i1 & i1 = i2 & (i1, v1) : sa..contents &
```

```
            0 <= i1 & i1 < sa..csize) | (sa..csize - 1 >= i1 & i1 > i2 & (i1, v1) :
            sa..contents & 0 <= i1 & i1 < sa..csize))" */
545
546         /*: assume "0 <= i2 & i2 < sb..csize" */
547         Object r2b = sb.remove_at(i2);
548         /*: assume "0 <= i1 & i1 <= sb..csize" */
549         sb.add_at(i1, v1);
550
551         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
552     }
553
554     static void add_at_remove_at_pre_s_15(ArrayList sa, ArrayList sb, int i1, Object
            v1, int i2)
555     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
556                   sa..contents = sb..contents & sa..csize = sb..csize"
557         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
558         ensures "True" */
559     {
560         /*: assume "(i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents) =
                ((i2, v) : sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2
                < sa..csize) | (sa..csize - 1 >= i1 & i1 = i2 & (i1, v1) : sa..contents & 0
                <= i1 & i1 < sa..csize) | (sa..csize - 1 >= i1 & i1 > i2 & (i1, v1) :
                sa..contents & 0 <= i1 & i1 < sa..csize)" */
561         /*: assume "0 <= i1 & i1 <= sa..csize" */
562         sa.add_at(i1, v1);
563         /*: assume "0 <= i2 & i2 < sa..csize" */
564         sa.remove_at(i2);
565
566         sb.remove_at(i2);
567         sb.add_at(i1, v1);
568
569         /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
570     }
571
572     static void add_at_remove_at_pre_c_15(ArrayList sa, ArrayList sb, int i1, Object
            v1, int i2)
573     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
574                   sa..contents = sb..contents & sa..csize = sb..csize"
575         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
576         ensures "True" */
577     {
578         /*: assume "~((i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents)
                = ((i2, v) : sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 &
                i2 < sa..csize) | (sa..csize - 1 >= i1 & i1 = i2 & (i1, v1) : sa..contents &
                0 <= i1 & i1 < sa..csize) | (sa..csize - 1 >= i1 & i1 > i2 & (i1, v1) :
                sa..contents & 0 <= i1 & i1 < sa..csize))" */
579         /*: assume "0 <= i1 & i1 <= sa..csize" */
580         sa.add_at(i1, v1);
581         /*: assume "0 <= i2 & i2 < sa..csize" */
582         sa.remove_at(i2);
583
584         /*: assume "0 <= i2 & i2 < sb..csize" */
585         sb.remove_at(i2);
586         /*: assume "0 <= i1 & i1 <= sb..csize" */
587         sb.add_at(i1, v1);
588
589         /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
590     }
591
592     static void add_at_remove_at_between_s_16(ArrayList sa, ArrayList sb, int i1,
            Object v1, int i2)
593     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
```

```
                sa..contents = sb..contents & sa..csize = sb..csize"
    modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
        "sb..msize"
    ensures "True" */
{
    /*: assume "0 <= i1 & i1 <= sa..csize" */
    sa.add_at(i1, v1);
    /*: assume "(i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa..contents) =
        ((i2 + 1, v) : sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2
        + 1 < sa..csize) | (sa..csize - 2 >= i1 & i1 = i2 & (i1 + 1, v1) :
        sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 2 >= i1 & i1
        > i2 & (i1 + 1, v1) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
    /*: assume "0 <= i2 & i2 < sa..csize" */
    sa.remove_at(i2);

    sb.remove_at(i2);
    sb.add_at(i1, v1);

    /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
}

static void add_at_remove_at_between_c_16(ArrayList sa, ArrayList sb, int i1,
    Object v1, int i2)
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
    modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
        "sb..msize"
    ensures "True" */
{
    /*: assume "0 <= i1 & i1 <= sa..csize" */
    sa.add_at(i1, v1);
    /*: assume "~((i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa..contents)
        = ((i2 + 1, v) : sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 &
        i2 + 1 < sa..csize) | (sa..csize - 2 >= i1 & i1 = i2 & (i1 + 1, v1) :
        sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 2 >= i1 & i1
        > i2 & (i1 + 1, v1) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
    /*: assume "0 <= i2 & i2 < sa..csize" */
    sa.remove_at(i2);

    /*: assume "0 <= i2 & i2 < sb..csize" */
    sb.remove_at(i2);
    /*: assume "0 <= i1 & i1 <= sb..csize" */
    sb.add_at(i1, v1);

    /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
}

static void add_at_remove_at_post_s_17(ArrayList sa, ArrayList sb, int i1, Object
    v1, int i2)
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
    modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
        "sb..msize"
    ensures "True" */
{
    /*: assume "0 <= i1 & i1 <= sa..csize" */
    sa.add_at(i1, v1);
    /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
    /*: ghost specvar sa__csize :: "int" = "sa..csize" */
    /*: assume "0 <= i2 & i2 < sa..csize" */
    sa.remove_at(i2);
    /*: assume "(i1 < i2 & i2 < sa..csize & (ALL v. ((i2, v) : sa__contents) =
        ((i2, v) : sa..contents)) & 0 <= i2 & i2 < sa__csize & 0 <= i2 & i2 <
        sa..csize) | (sa..csize - 1 >= i1 & i1 = i2 & (i1, v1) : sa..contents & 0 <=
```

```
        i1 & i1 < sa..csize) | (sa..csize - 1 >= i1 & i1 > i2 & (i1, v1) :
        sa..contents & 0 <= i1 & i1 < sa..csize)" */

    sb.remove_at(i2);
    sb.add_at(i1, v1);

    /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
}

static void add_at_remove_at_post_c_17(ArrayList sa, ArrayList sb, int i1, Object
    v1, int i2)
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
              sa..contents = sb..contents & sa..csize = sb..csize"
    modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
        "sb..msize"
    ensures "True" */
{
    /*: assume "0 <= i1 & i1 <= sa..csize" */
    sa.add_at(i1, v1);
    /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
    /*: ghost specvar sa__csize :: "int" = "sa..csize" */
    /*: assume "0 <= i2 & i2 < sa..csize" */
    sa.remove_at(i2);
    /*: assume "~((i1 < i2 & i2 < sa..csize & (ALL v. ((i2, v) : sa__contents) =
        ((i2, v) : sa..contents)) & 0 <= i2 & i2 < sa__csize & 0 <= i2 & i2 <
        sa..csize) | (sa..csize - 1 >= i1 & i1 = i2 & (i1, v1) : sa..contents & 0 <=
        i1 & i1 < sa..csize) | (sa..csize - 1 >= i1 & i1 > i2 & (i1, v1) :
        sa..contents & 0 <= i1 & i1 < sa..csize))" */

    /*: assume "0 <= i2 & i2 < sb..csize" */
    sb.remove_at(i2);
    /*: assume "0 <= i1 & i1 <= sb..csize" */
    sb.add_at(i1, v1);

    /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
}

static void add_at_set_pre_s_18(ArrayList sa, ArrayList sb, int i1, Object v1, int
    i2, Object v2)
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
              sa..contents = sb..contents & sa..csize = sb..csize"
    modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
        "sb..msize"
    ensures "True" */
{
    /*: assume "(i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents) =
        ((i2, v) : sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) :
        sa..contents & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize)
        | (i1 = i2 & i2 < sa..csize & (i2, v1) : sa..contents & (i2, v2) :
        sa..contents & v1 = v2 & 0 <= i2 & i2 < sa..csize) | i1 > i2" */
    /*: assume "0 <= i1 & i1 <= sa..csize" */
    sa.add_at(i1, v1);
    /*: assume "0 <= i2 & i2 < sa..csize" */
    Object r2a = sa.set(i2, v2);

    Object r2b = sb.set(i2, v2);
    sb.add_at(i1, v1);

    /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
}

static void add_at_set_pre_c_18(ArrayList sa, ArrayList sb, int i1, Object v1, int
    i2, Object v2)
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
              sa..contents = sb..contents & sa..csize = sb..csize"
```

```
693         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
694         ensures "True" */
695     {
696         /*: assume "~((i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents)
                = ((i2, v) : sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) :
                sa..contents & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize)
                | (i1 = i2 & i2 < sa..csize & (i2, v1) : sa..contents & (i2, v2) :
                sa..contents & v1 = v2 & 0 <= i2 & i2 < sa..csize) | i1 > i2)" */
697         /*: assume "0 <= i1 & i1 <= sa..csize" */
698         sa.add_at(i1, v1);
699         /*: assume "0 <= i2 & i2 < sa..csize" */
700         Object r2a = sa.set(i2, v2);
701 
702         /*: assume "0 <= i2 & i2 < sb..csize" */
703         Object r2b = sb.set(i2, v2);
704         /*: assume "0 <= i1 & i1 <= sb..csize" */
705         sb.add_at(i1, v1);
706 
707         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
708     }
709 
710     static void add_at_set_between_s_19(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2, Object v2)
711     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
712                   sa..contents = sb..contents & sa..csize = sb..csize"
713         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
714         ensures "True" */
715     {
716         /*: assume "0 <= i1 & i1 <= sa..csize" */
717         sa.add_at(i1, v1);
718         /*: assume "(i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa..contents) =
                ((i2 + 1, v) : sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) :
                sa..contents & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize)
                | (i1 = i2 & i2 < sa..csize - 1 & (i2 + 1, v1) : sa..contents & (i2 + 1, v2)
                : sa..contents & v1 = v2 & 0 <= i2 + 1 & i2 + 1 < sa..csize) | i1 > i2" */
719         /*: assume "0 <= i2 & i2 < sa..csize" */
720         Object r2a = sa.set(i2, v2);
721 
722         Object r2b = sb.set(i2, v2);
723         sb.add_at(i1, v1);
724 
725         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
726     }
727 
728     static void add_at_set_between_c_19(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2, Object v2)
729     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
730                   sa..contents = sb..contents & sa..csize = sb..csize"
731         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
732         ensures "True" */
733     {
734         /*: assume "0 <= i1 & i1 <= sa..csize" */
735         sa.add_at(i1, v1);
736         /*: assume "~((i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa..contents)
                = ((i2 + 1, v) : sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) :
                sa..contents & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize)
                | (i1 = i2 & i2 < sa..csize - 1 & (i2 + 1, v1) : sa..contents & (i2 + 1, v2)
                : sa..contents & v1 = v2 & 0 <= i2 + 1 & i2 + 1 < sa..csize) | i1 > i2)" */
737         /*: assume "0 <= i2 & i2 < sa..csize" */
738         Object r2a = sa.set(i2, v2);
739 
```

```
740         /*: assume "0 <= i2 & i2 < sb..csize" */
741         Object r2b = sb.set(i2, v2);
742         /*: assume "0 <= i1 & i1 <= sb..csize" */
743         sb.add_at(i1, v1);
744 
745         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
746     }
747 
748     static void add_at_set_post_s_20(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2, Object v2)
749     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
750                 sa..contents = sb..contents & sa..csize = sb..csize"
751         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
752         ensures "True" */
753     {
754         /*: assume "0 <= i1 & i1 <= sa..csize" */
755         sa.add_at(i1, v1);
756         /*: assume "0 <= i2 & i2 < sa..csize" */
757         Object r2a = sa.set(i2, v2);
758         /*: assume "(i1 < i2 & i2 < sa..csize - 1 & (i2 + 1, r2a) : sa..contents & r2a
                = v2 & (i2 + 1, v2) : sa..contents & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1
                = i2 & i2 < sa..csize - 1 & (i2 + 1, v1) : sa..contents & (i2 + 1, v2) :
                sa..contents & v1 = v2 & 0 <= i2 + 1 & i2 + 1 < sa..csize) | i1 > i2" */
759 
760         Object r2b = sb.set(i2, v2);
761         sb.add_at(i1, v1);
762 
763         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
764     }
765 
766     static void add_at_set_post_c_20(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2, Object v2)
767     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
768                 sa..contents = sb..contents & sa..csize = sb..csize"
769         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
770         ensures "True" */
771     {
772         /*: assume "0 <= i1 & i1 <= sa..csize" */
773         sa.add_at(i1, v1);
774         /*: assume "0 <= i2 & i2 < sa..csize" */
775         Object r2a = sa.set(i2, v2);
776         /*: assume "~((i1 < i2 & i2 < sa..csize - 1 & (i2 + 1, r2a) : sa..contents &
                r2a = v2 & (i2 + 1, v2) : sa..contents & 0 <= i2 + 1 & i2 + 1 < sa..csize) |
                (i1 = i2 & i2 < sa..csize - 1 & (i2 + 1, v1) : sa..contents & (i2 + 1, v2) :
                sa..contents & v1 = v2 & 0 <= i2 + 1 & i2 + 1 < sa..csize) | i1 > i2)" */
777 
778         /*: assume "0 <= i2 & i2 < sb..csize" */
779         Object r2b = sb.set(i2, v2);
780         /*: assume "0 <= i1 & i1 <= sb..csize" */
781         sb.add_at(i1, v1);
782 
783         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
784     }
785 
786     static void add_at_set_pre_s_21(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2, Object v2)
787     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
788                 sa..contents = sb..contents & sa..csize = sb..csize"
789         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
790         ensures "True" */
```

```
791     {
792         /*: assume "(i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents) =
                ((i2, v) : sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) :
                sa..contents & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize)
                | (i1 = i2 & i2 < sa..csize & (i2, v1) : sa..contents & (i2, v2) :
                sa..contents & v1 = v2 & 0 <= i2 & i2 < sa..csize) | i1 > i2" */
793         /*: assume "0 <= i1 & i1 <= sa..csize" */
794         sa.add_at(i1, v1);
795         /*: assume "0 <= i2 & i2 < sa..csize" */
796         sa.set(i2, v2);
797 
798         sb.set(i2, v2);
799         sb.add_at(i1, v1);
800 
801         /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
802     }
803 
804     static void add_at_set_pre_c_21(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2, Object v2)
805     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
806                   sa..contents = sb..contents & sa..csize = sb..csize"
807         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
808         ensures "True" */
809     {
810         /*: assume "~((i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents)
                = ((i2, v) : sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) :
                sa..contents & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize)
                | (i1 = i2 & i2 < sa..csize & (i2, v1) : sa..contents & (i2, v2) :
                sa..contents & v1 = v2 & 0 <= i2 & i2 < sa..csize) | i1 > i2)" */
811         /*: assume "0 <= i1 & i1 <= sa..csize" */
812         sa.add_at(i1, v1);
813         /*: assume "0 <= i2 & i2 < sa..csize" */
814         sa.set(i2, v2);
815 
816         /*: assume "0 <= i2 & i2 < sb..csize" */
817         sb.set(i2, v2);
818         /*: assume "0 <= i1 & i1 <= sb..csize" */
819         sb.add_at(i1, v1);
820 
821         /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
822     }
823 
824     static void add_at_set_between_s_22(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2, Object v2)
825     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
826                   sa..contents = sb..contents & sa..csize = sb..csize"
827         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
828         ensures "True" */
829     {
830         /*: assume "0 <= i1 & i1 <= sa..csize" */
831         sa.add_at(i1, v1);
832         /*: assume "(i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa..contents) =
                ((i2 + 1, v) : sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) :
                sa..contents & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize)
                | (i1 = i2 & i2 < sa..csize - 1 & (i2 + 1, v1) : sa..contents & (i2 + 1, v2)
                : sa..contents & v1 = v2 & 0 <= i2 + 1 & i2 + 1 < sa..csize) | i1 > i2" */
833         /*: assume "0 <= i2 & i2 < sa..csize" */
834         sa.set(i2, v2);
835 
836         sb.set(i2, v2);
837         sb.add_at(i1, v1);
838 
839         /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
```

```
840     }
841
842     static void add_at_set_between_c_22(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2, Object v2)
843     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
844                     sa..contents = sb..contents & sa..csize = sb..csize"
845         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
846         ensures "True" */
847     {
848         /*: assume "0 <= i1 & i1 <= sa..csize" */
849         sa.add_at(i1, v1);
850         /*: assume "~((i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa..contents)
                = ((i2 + 1, v) : sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) :
                sa..contents & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize)
                | (i1 = i2 & i2 < sa..csize - 1 & (i2 + 1, v1) : sa..contents & (i2 + 1, v2)
                : sa..contents & v1 = v2 & 0 <= i2 + 1 & i2 + 1 < sa..csize) | i1 > i2)" */
851         /*: assume "0 <= i2 & i2 < sa..csize" */
852         sa.set(i2, v2);
853
854         /*: assume "0 <= i2 & i2 < sb..csize" */
855         sb.set(i2, v2);
856         /*: assume "0 <= i1 & i1 <= sb..csize" */
857         sb.add_at(i1, v1);
858
859         /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
860     }
861
862     static void add_at_set_post_s_23(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2, Object v2)
863     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
864                     sa..contents = sb..contents & sa..csize = sb..csize"
865         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
866         ensures "True" */
867     {
868         /*: assume "0 <= i1 & i1 <= sa..csize" */
869         sa.add_at(i1, v1);
870         /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
871         /*: ghost specvar sa__csize :: "int" = "sa..csize" */
872         /*: assume "0 <= i2 & i2 < sa..csize" */
873         sa.set(i2, v2);
874         /*: assume "(i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa__contents) =
                ((i2 + 1, v) : sa..contents)) & (i2, v2) : sa__contents & (i2 + 1, v2) :
                sa..contents & 0 <= i2 & i2 < sa__csize & 0 <= i2 + 1 & i2 + 1 < sa..csize)
                | (i1 = i2 & i2 < sa..csize - 1 & (i2 + 1, v1) : sa..contents & (i2 + 1, v2)
                : sa..contents & v1 = v2 & 0 <= i2 + 1 & i2 + 1 < sa..csize) | i1 > i2" */
875
876         sb.set(i2, v2);
877         sb.add_at(i1, v1);
878
879         /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
880     }
881
882     static void add_at_set_post_c_23(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2, Object v2)
883     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
884                     sa..contents = sb..contents & sa..csize = sb..csize"
885         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
886         ensures "True" */
887     {
888         /*: assume "0 <= i1 & i1 <= sa..csize" */
889         sa.add_at(i1, v1);
890         /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
```

```
        /*: ghost specvar sa__csize :: "int" = "sa..csize" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);
        /*: assume "~((i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa__contents)
            = ((i2 + 1, v) : sa..contents)) & (i2, v2) : sa__contents & (i2 + 1, v2) :
            sa..contents & 0 <= i2 & i2 < sa__csize & 0 <= i2 + 1 & i2 + 1 < sa..csize)
            | (i1 = i2 & i2 < sa..csize - 1 & (i2 + 1, v1) : sa..contents & (i2 + 1, v2)
            : sa..contents & v1 = v2 & 0 <= i2 + 1 & i2 + 1 < sa..csize) | i1 > i2)" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 <= sb..csize" */
        sb.add_at(i1, v1);

        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
    }

    static void add_at_size_pre_s_24(ArrayList sa, ArrayList sb, int i1, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "False" */
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        int r2a = sa.size();

        int r2b = sb.size();
        sb.add_at(i1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void add_at_size_pre_c_24(ArrayList sa, ArrayList sb, int i1, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "~(False)" */
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        int r2a = sa.size();

        int r2b = sb.size();
        /*: assume "0 <= i1 & i1 <= sb..csize" */
        sb.add_at(i1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void add_at_size_between_s_25(ArrayList sa, ArrayList sb, int i1, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "False" */
```

```
948         int r2a = sa.size();
949 
950         int r2b = sb.size();
951         sb.add_at(i1, v1);
952 
953         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
954     }
955 
956     static void add_at_size_between_c_25(ArrayList sa, ArrayList sb, int i1, Object v1)
957     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
958                   sa..contents = sb..contents & sa..csize = sb..csize"
959         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
960         ensures "True" */
961     {
962         /*: assume "0 <= i1 & i1 <= sa..csize" */
963         sa.add_at(i1, v1);
964         /*: assume "~(False)" */
965         int r2a = sa.size();
966 
967         int r2b = sb.size();
968         /*: assume "0 <= i1 & i1 <= sb..csize" */
969         sb.add_at(i1, v1);
970 
971         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
972     }
973 
974     static void add_at_size_post_s_26(ArrayList sa, ArrayList sb, int i1, Object v1)
975     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
976                   sa..contents = sb..contents & sa..csize = sb..csize"
977         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
978         ensures "True" */
979     {
980         /*: assume "0 <= i1 & i1 <= sa..csize" */
981         sa.add_at(i1, v1);
982         int r2a = sa.size();
983         /*: assume "False" */
984 
985         int r2b = sb.size();
986         sb.add_at(i1, v1);
987 
988         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
989     }
990 
991     static void add_at_size_post_c_26(ArrayList sa, ArrayList sb, int i1, Object v1)
992     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
993                   sa..contents = sb..contents & sa..csize = sb..csize"
994         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
995         ensures "True" */
996     {
997         /*: assume "0 <= i1 & i1 <= sa..csize" */
998         sa.add_at(i1, v1);
999         int r2a = sa.size();
1000        /*: assume "~(False)" */
1001 
1002        int r2b = sb.size();
1003        /*: assume "0 <= i1 & i1 <= sb..csize" */
1004        sb.add_at(i1, v1);
1005 
1006        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
1007    }
```

```
    static void get_add_at_pre_s_27(ArrayList sa, ArrayList sb, int i1, int i2, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "i1 < i2 | (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 <
            sa..csize) | (i1 > i2 & (ALL v. ((i1 - 1, v) : sa..contents) = ((i1, v) :
            sa..contents)) & 0 <= i1 - 1 & i1 - 1 < sa..csize & 0 <= i1 & i1 <
            sa..csize)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        sb.add_at(i2, v2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void get_add_at_pre_c_27(ArrayList sa, ArrayList sb, int i1, int i2, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 <
            sa..csize) | (i1 > i2 & (ALL v. ((i1 - 1, v) : sa..contents) = ((i1, v) :
            sa..contents)) & 0 <= i1 - 1 & i1 - 1 < sa..csize & 0 <= i1 & i1 <
            sa..csize))" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        /*: assume "0 <= i2 & i2 <= sb..csize" */
        sb.add_at(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void get_add_at_between_s_28(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "i1 < i2 | (i1 = i2 & r1a = v2) | (i1 > i2 & (i1 - 1, r1a) :
            sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize)" */
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);
```

```
1059        sb.add_at(i2, v2);
1060        Object r1b = sb.get(i1);
1061
1062        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
1063    }
1064
1065    static void get_add_at_between_c_28(ArrayList sa, ArrayList sb, int i1, int i2,
            Object v2)
1066    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1067                  sa..contents = sb..contents & sa..csize = sb..csize"
1068        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
1069        ensures "True" */
1070    {
1071        /*: assume "0 <= i1 & i1 < sa..csize" */
1072        Object r1a = sa.get(i1);
1073        /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v2) | (i1 > i2 & (i1 - 1, r1a) :
                sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize))" */
1074        /*: assume "0 <= i2 & i2 <= sa..csize" */
1075        sa.add_at(i2, v2);
1076
1077        /*: assume "0 <= i2 & i2 <= sb..csize" */
1078        sb.add_at(i2, v2);
1079        /*: assume "0 <= i1 & i1 < sb..csize" */
1080        Object r1b = sb.get(i1);
1081
1082        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
1083    }
1084
1085    static void get_add_at_post_s_29(ArrayList sa, ArrayList sb, int i1, int i2, Object
            v2)
1086    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1087                  sa..contents = sb..contents & sa..csize = sb..csize"
1088        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
1089        ensures "True" */
1090    {
1091        /*: assume "0 <= i1 & i1 < sa..csize" */
1092        Object r1a = sa.get(i1);
1093        /*: assume "0 <= i2 & i2 <= sa..csize" */
1094        sa.add_at(i2, v2);
1095        /*: assume "i1 < i2 | (i1 = i2 & r1a = v2) | (i1 > i2 & (i1, r1a) :
                sa..contents & 0 <= i1 & i1 < sa..csize)" */
1096
1097        sb.add_at(i2, v2);
1098        Object r1b = sb.get(i1);
1099
1100        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
1101    }
1102
1103    static void get_add_at_post_c_29(ArrayList sa, ArrayList sb, int i1, int i2, Object
            v2)
1104    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1105                  sa..contents = sb..contents & sa..csize = sb..csize"
1106        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
1107        ensures "True" */
1108    {
1109        /*: assume "0 <= i1 & i1 < sa..csize" */
1110        Object r1a = sa.get(i1);
1111        /*: assume "0 <= i2 & i2 <= sa..csize" */
1112        sa.add_at(i2, v2);
1113        /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v2) | (i1 > i2 & (i1, r1a) :
                sa..contents & 0 <= i1 & i1 < sa..csize))" */
```

```
1114
1115        /*: assume "0 <= i2 & i2 <= sb..csize" */
1116        sb.add_at(i2, v2);
1117        /*: assume "0 <= i1 & i1 < sb..csize" */
1118        Object r1b = sb.get(i1);
1119
1120        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
1121    }
1122
1123    static void get_get_pre_s_30(ArrayList sa, ArrayList sb, int i1, int i2)
1124    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1125                  sa..contents = sb..contents & sa..csize = sb..csize"
1126        ensures "True" */
1127    {
1128        /*: assume "True" */
1129        /*: assume "0 <= i1 & i1 < sa..csize" */
1130        Object r1a = sa.get(i1);
1131        /*: assume "0 <= i2 & i2 < sa..csize" */
1132        Object r2a = sa.get(i2);
1133
1134        Object r2b = sb.get(i2);
1135        Object r1b = sb.get(i1);
1136
1137        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
1138    }
1139
1140    static void get_get_pre_c_30(ArrayList sa, ArrayList sb, int i1, int i2)
1141    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1142                  sa..contents = sb..contents & sa..csize = sb..csize"
1143        ensures "True" */
1144    {
1145        /*: assume "~(True)" */
1146        /*: assume "0 <= i1 & i1 < sa..csize" */
1147        Object r1a = sa.get(i1);
1148        /*: assume "0 <= i2 & i2 < sa..csize" */
1149        Object r2a = sa.get(i2);
1150
1151        /*: assume "0 <= i2 & i2 < sb..csize" */
1152        Object r2b = sb.get(i2);
1153        /*: assume "0 <= i1 & i1 < sb..csize" */
1154        Object r1b = sb.get(i1);
1155
1156        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
1157    }
1158
1159    static void get_get_between_s_31(ArrayList sa, ArrayList sb, int i1, int i2)
1160    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1161                  sa..contents = sb..contents & sa..csize = sb..csize"
1162        ensures "True" */
1163    {
1164        /*: assume "0 <= i1 & i1 < sa..csize" */
1165        Object r1a = sa.get(i1);
1166        /*: assume "True" */
1167        /*: assume "0 <= i2 & i2 < sa..csize" */
1168        Object r2a = sa.get(i2);
1169
1170        Object r2b = sb.get(i2);
1171        Object r1b = sb.get(i1);
1172
1173        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
1174    }
```

```
1175
1176    static void get_get_between_c_31(ArrayList sa, ArrayList sb, int i1, int i2)
1177    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1178                sa..contents = sb..contents & sa..csize = sb..csize"
1179        ensures "True" */
1180    {
1181        /*: assume "0 <= i1 & i1 < sa..csize" */
1182        Object r1a = sa.get(i1);
1183        /*: assume "~(True)" */
1184        /*: assume "0 <= i2 & i2 < sa..csize" */
1185        Object r2a = sa.get(i2);
1186
1187        /*: assume "0 <= i2 & i2 < sb..csize" */
1188        Object r2b = sb.get(i2);
1189        /*: assume "0 <= i1 & i1 < sb..csize" */
1190        Object r1b = sb.get(i1);
1191
1192        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
1193    }
1194
1195    static void get_get_post_s_32(ArrayList sa, ArrayList sb, int i1, int i2)
1196    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1197                sa..contents = sb..contents & sa..csize = sb..csize"
1198        ensures "True" */
1199    {
1200        /*: assume "0 <= i1 & i1 < sa..csize" */
1201        Object r1a = sa.get(i1);
1202        /*: assume "0 <= i2 & i2 < sa..csize" */
1203        Object r2a = sa.get(i2);
1204        /*: assume "True" */
1205
1206        Object r2b = sb.get(i2);
1207        Object r1b = sb.get(i1);
1208
1209        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
1210    }
1211
1212    static void get_get_post_c_32(ArrayList sa, ArrayList sb, int i1, int i2)
1213    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1214                sa..contents = sb..contents & sa..csize = sb..csize"
1215        ensures "True" */
1216    {
1217        /*: assume "0 <= i1 & i1 < sa..csize" */
1218        Object r1a = sa.get(i1);
1219        /*: assume "0 <= i2 & i2 < sa..csize" */
1220        Object r2a = sa.get(i2);
1221        /*: assume "~(True)" */
1222
1223        /*: assume "0 <= i2 & i2 < sb..csize" */
1224        Object r2b = sb.get(i2);
1225        /*: assume "0 <= i1 & i1 < sb..csize" */
1226        Object r1b = sb.get(i1);
1227
1228        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
1229    }
1230
1231    static void get_indexOf_pre_s_33(ArrayList sa, ArrayList sb, int i1, Object v2)
1232    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1233                sa..contents = sb..contents & sa..csize = sb..csize"
1234        ensures "True" */
1235    {
1236        /*: assume "True" */
```

```
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        int r2a = sa.indexOf(v2);

        int r2b = sb.indexOf(v2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void get_indexOf_pre_c_33(ArrayList sa, ArrayList sb, int i1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "~(True)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        int r2a = sa.indexOf(v2);

        int r2b = sb.indexOf(v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void get_indexOf_between_s_34(ArrayList sa, ArrayList sb, int i1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "True" */
        int r2a = sa.indexOf(v2);

        int r2b = sb.indexOf(v2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void get_indexOf_between_c_34(ArrayList sa, ArrayList sb, int i1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "~(True)" */
        int r2a = sa.indexOf(v2);

        int r2b = sb.indexOf(v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void get_indexOf_post_s_35(ArrayList sa, ArrayList sb, int i1, Object v2)
```

```
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        int r2a = sa.indexOf(v2);
        /*: assume "True" */

        int r2b = sb.indexOf(v2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void get_indexOf_post_c_35(ArrayList sa, ArrayList sb, int i1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        int r2a = sa.indexOf(v2);
        /*: assume "~(True)" */

        int r2b = sb.indexOf(v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void get_lastIndexOf_pre_s_36(ArrayList sa, ArrayList sb, int i1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "True" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        int r2a = sa.lastIndexOf(v2);

        int r2b = sb.lastIndexOf(v2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void get_lastIndexOf_pre_c_36(ArrayList sa, ArrayList sb, int i1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "~(True)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        int r2a = sa.lastIndexOf(v2);

        int r2b = sb.lastIndexOf(v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

```

```
        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void get_lastIndexOf_between_s_37(ArrayList sa, ArrayList sb, int i1, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "True" */
        int r2a = sa.lastIndexOf(v2);

        int r2b = sb.lastIndexOf(v2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void get_lastIndexOf_between_c_37(ArrayList sa, ArrayList sb, int i1, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "~(True)" */
        int r2a = sa.lastIndexOf(v2);

        int r2b = sb.lastIndexOf(v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void get_lastIndexOf_post_s_38(ArrayList sa, ArrayList sb, int i1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        int r2a = sa.lastIndexOf(v2);
        /*: assume "True" */

        int r2b = sb.lastIndexOf(v2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void get_lastIndexOf_post_c_38(ArrayList sa, ArrayList sb, int i1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
```

```
        int r2a = sa.lastIndexOf(v2);
        /*: assume "~(True)" */

        int r2b = sb.lastIndexOf(v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void get_remove_at_pre_s_39(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "i1 < i2 | (sa..csize - 1 > i1 & i1 = i2 & (ALL v. ((i1, v) :
            sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
            <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v.
            ((i1, v) : sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 <
            sa..csize & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);

        Object r2b = sb.remove_at(i2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void get_remove_at_pre_c_39(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~(i1 < i2 | (sa..csize - 1 > i1 & i1 = i2 & (ALL v. ((i1, v) :
            sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
            <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v.
            ((i1, v) : sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 <
            sa..csize & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void get_remove_at_between_s_40(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
```

```
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "i1 < i2 | (sa..csize - 1 > i1 & i1 = i2 & (i1 + 1, r1a) :
            sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
            > i2 & (i1 + 1, r1a) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);

        Object r2b = sb.remove_at(i2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void get_remove_at_between_c_40(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "~(i1 < i2 | (sa..csize - 1 > i1 & i1 = i2 & (i1 + 1, r1a) :
            sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
            > i2 & (i1 + 1, r1a) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void get_remove_at_post_s_41(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);
        /*: assume "i1 < i2 | (sa..csize > i1 & i1 = i2 & (i1, r1a) : sa..contents & 0
            <= i1 & i1 < sa..csize) | (sa..csize > i1 & i1 > i2 & (i1, r1a) :
            sa..contents & 0 <= i1 & i1 < sa..csize)" */

        Object r2b = sb.remove_at(i2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void get_remove_at_post_c_41(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
```

```
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);
        /*: assume "~(i1 < i2 | (sa..csize > i1 & i1 = i2 & (i1, r1a) : sa..contents &
            0 <= i1 & i1 < sa..csize) | (sa..csize > i1 & i1 > i2 & (i1, r1a) :
            sa..contents & 0 <= i1 & i1 < sa..csize))" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void get_remove_at_pre_s_42(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "i1 < i2 | (sa..csize - 1 > i1 & i1 = i2 & (ALL v. ((i1, v) :
            sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
            <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v.
            ((i1, v) : sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 <
            sa..csize & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        sb.remove_at(i2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void get_remove_at_pre_c_42(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~(i1 < i2 | (sa..csize - 1 > i1 & i1 = i2 & (ALL v. ((i1, v) :
            sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
            <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v.
            ((i1, v) : sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 <
            sa..csize & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void get_remove_at_between_s_43(ArrayList sa, ArrayList sb, int i1, int i2)
```

```
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "i1 < i2 | (sa..csize - 1 > i1 & i1 = i2 & (i1 + 1, r1a) :
            sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
            > i2 & (i1 + 1, r1a) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        sb.remove_at(i2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void get_remove_at_between_c_43(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "~(i1 < i2 | (sa..csize - 1 > i1 & i1 = i2 & (i1 + 1, r1a) :
            sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
            > i2 & (i1 + 1, r1a) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void get_remove_at_post_s_44(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);
        /*: assume "i1 < i2 | (sa..csize > i1 & i1 = i2 & (i1, r1a) : sa..contents & 0
            <= i1 & i1 < sa..csize) | (sa..csize > i1 & i1 > i2 & (i1, r1a) :
            sa..contents & 0 <= i1 & i1 < sa..csize)" */

        sb.remove_at(i2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void get_remove_at_post_c_44(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
```

```
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);
        /*: assume "~(i1 < i2 | (sa..csize > i1 & i1 = i2 & (i1, r1a) : sa..contents &
            0 <= i1 & i1 < sa..csize) | (sa..csize > i1 & i1 > i2 & (i1, r1a) :
            sa..contents & 0 <= i1 & i1 < sa..csize))" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void get_set_pre_s_45(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "i1 < i2 | (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 <
            sa..csize) | i1 > i2" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        Object r2b = sb.set(i2, v2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void get_set_pre_c_45(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 <
            sa..csize) | i1 > i2)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void get_set_between_s_46(ArrayList sa, ArrayList sb, int i1, int i2, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
```

```
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        Object r2b = sb.set(i2, v2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void get_set_between_c_46(ArrayList sa, ArrayList sb, int i1, int i2, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void get_set_post_s_47(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);
        /*: assume "i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2" */

        Object r2b = sb.set(i2, v2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void get_set_post_c_47(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
```

```
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);
        /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2)" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void get_set_pre_s_48(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "i1 < i2 | (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 <
            sa..csize) | i1 > i2" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        sb.set(i2, v2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void get_set_pre_c_48(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 <
            sa..csize) | i1 > i2)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void get_set_between_s_49(ArrayList sa, ArrayList sb, int i1, int i2, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2" */
```

```
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        sb.set(i2, v2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void get_set_between_c_49(ArrayList sa, ArrayList sb, int i1, int i2, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void get_set_post_s_50(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);
        /*: assume "i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2" */

        sb.set(i2, v2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void get_set_post_c_50(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);
        /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2)" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);
```

```
1881
1882        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
1883    }
1884
1885    static void get_size_pre_s_51(ArrayList sa, ArrayList sb, int i1)
1886    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1887                  sa..contents = sb..contents & sa..csize = sb..csize"
1888        ensures "True" */
1889    {
1890        /*: assume "True" */
1891        /*: assume "0 <= i1 & i1 < sa..csize" */
1892        Object r1a = sa.get(i1);
1893        int r2a = sa.size();
1894
1895        int r2b = sb.size();
1896        Object r1b = sb.get(i1);
1897
1898        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
1899    }
1900
1901    static void get_size_pre_c_51(ArrayList sa, ArrayList sb, int i1)
1902    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1903                  sa..contents = sb..contents & sa..csize = sb..csize"
1904        ensures "True" */
1905    {
1906        /*: assume "~(True)" */
1907        /*: assume "0 <= i1 & i1 < sa..csize" */
1908        Object r1a = sa.get(i1);
1909        int r2a = sa.size();
1910
1911        int r2b = sb.size();
1912        /*: assume "0 <= i1 & i1 < sb..csize" */
1913        Object r1b = sb.get(i1);
1914
1915        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
1916    }
1917
1918    static void get_size_between_s_52(ArrayList sa, ArrayList sb, int i1)
1919    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1920                  sa..contents = sb..contents & sa..csize = sb..csize"
1921        ensures "True" */
1922    {
1923        /*: assume "0 <= i1 & i1 < sa..csize" */
1924        Object r1a = sa.get(i1);
1925        /*: assume "True" */
1926        int r2a = sa.size();
1927
1928        int r2b = sb.size();
1929        Object r1b = sb.get(i1);
1930
1931        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
1932    }
1933
1934    static void get_size_between_c_52(ArrayList sa, ArrayList sb, int i1)
1935    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1936                  sa..contents = sb..contents & sa..csize = sb..csize"
1937        ensures "True" */
1938    {
1939        /*: assume "0 <= i1 & i1 < sa..csize" */
1940        Object r1a = sa.get(i1);
1941        /*: assume "~(True)" */
```

```
1942        int r2a = sa.size();
1943
1944        int r2b = sb.size();
1945        /*: assume "0 <= i1 & i1 < sb..csize" */
1946        Object r1b = sb.get(i1);
1947
1948        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
1949    }
1950
1951    static void get_size_post_s_53(ArrayList sa, ArrayList sb, int i1)
1952    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1953                  sa..contents = sb..contents & sa..csize = sb..csize"
1954        ensures "True" */
1955    {
1956        /*: assume "0 <= i1 & i1 < sa..csize" */
1957        Object r1a = sa.get(i1);
1958        int r2a = sa.size();
1959        /*: assume "True" */
1960
1961        int r2b = sb.size();
1962        Object r1b = sb.get(i1);
1963
1964        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
1965    }
1966
1967    static void get_size_post_c_53(ArrayList sa, ArrayList sb, int i1)
1968    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1969                  sa..contents = sb..contents & sa..csize = sb..csize"
1970        ensures "True" */
1971    {
1972        /*: assume "0 <= i1 & i1 < sa..csize" */
1973        Object r1a = sa.get(i1);
1974        int r2a = sa.size();
1975        /*: assume "~(True)" */
1976
1977        int r2b = sb.size();
1978        /*: assume "0 <= i1 & i1 < sb..csize" */
1979        Object r1b = sb.get(i1);
1980
1981        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
1982    }
1983
1984    static void indexOf_add_at_pre_s_54(ArrayList sa, ArrayList sb, Object v1, int i2,
            Object v2)
1985    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1986                  sa..contents = sb..contents & sa..csize = sb..csize"
1987        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
1988        ensures "True" */
1989    {
1990        /*: assume "(~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
                v2) | (EX i. (i, v1) : sa..contents & 0 <= i & i < i2) | (~(EX i. (i, v1) :
                sa..contents & 0 <= i & i < i2) & (i2, v1) : sa..contents & 0 <= i2 & i2 <
                sa..csize & v1 = v2)" */
1991        int r1a = sa.indexOf(v1);
1992        /*: assume "0 <= i2 & i2 <= sa..csize" */
1993        sa.add_at(i2, v2);
1994
1995        sb.add_at(i2, v2);
1996        int r1b = sb.indexOf(v1);
1997
1998        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
```

```
    }

    static void indexOf_add_at_pre_c_54(ArrayList sa, ArrayList sb, Object v1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "~((~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
            v2) | (EX i. (i, v1) : sa..contents & 0 <= i & i < i2) | (~(EX i. (i, v1) :
            sa..contents & 0 <= i & i < i2) & (i2, v1) : sa..contents & 0 <= i2 & i2 <
            sa..csize & v1 = v2))" */
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        /*: assume "0 <= i2 & i2 <= sb..csize" */
        sb.add_at(i2, v2);
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void indexOf_add_at_between_s_55(ArrayList sa, ArrayList sb, Object v1, int
        i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 =
            v2)" */
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        sb.add_at(i2, v2);
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void indexOf_add_at_between_c_55(ArrayList sa, ArrayList sb, Object v1, int
        i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 =
            v2))" */
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        /*: assume "0 <= i2 & i2 <= sb..csize" */
        sb.add_at(i2, v2);
        int r1b = sb.indexOf(v1);
```

```
        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void indexOf_add_at_post_s_56(ArrayList sa, ArrayList sb, Object v1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);
        /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 =
            v2)" */

        sb.add_at(i2, v2);
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void indexOf_add_at_post_c_56(ArrayList sa, ArrayList sb, Object v1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);
        /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 =
            v2))" */

        /*: assume "0 <= i2 & i2 <= sb..csize" */
        sb.add_at(i2, v2);
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void indexOf_get_pre_s_57(ArrayList sa, ArrayList sb, Object v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "True" */
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        Object r2b = sb.get(i2);
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void indexOf_get_pre_c_57(ArrayList sa, ArrayList sb, Object v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
```

```
                sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "~(True)" */
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.get(i2);
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void indexOf_get_between_s_58(ArrayList sa, ArrayList sb, Object v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "True" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        Object r2b = sb.get(i2);
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void indexOf_get_between_c_58(ArrayList sa, ArrayList sb, Object v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "~(True)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.get(i2);
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void indexOf_get_post_s_59(ArrayList sa, ArrayList sb, Object v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);
        /*: assume "True" */

        Object r2b = sb.get(i2);
        int r1b = sb.indexOf(v1);

```

```
2168        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
2169    }
2170
2171    static void indexOf_get_post_c_59(ArrayList sa, ArrayList sb, Object v1, int i2)
2172    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2173                  sa..contents = sb..contents & sa..csize = sb..csize"
2174        ensures "True" */
2175    {
2176        int r1a = sa.indexOf(v1);
2177        /*: assume "0 <= i2 & i2 < sa..csize" */
2178        Object r2a = sa.get(i2);
2179        /*: assume "~(True)" */
2180
2181        /*: assume "0 <= i2 & i2 < sb..csize" */
2182        Object r2b = sb.get(i2);
2183        int r1b = sb.indexOf(v1);
2184
2185        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
2186    }
2187
2188    static void indexOf_indexOf_pre_s_60(ArrayList sa, ArrayList sb, Object v1, Object
            v2)
2189    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2190                  sa..contents = sb..contents & sa..csize = sb..csize"
2191        ensures "True" */
2192    {
2193        /*: assume "True" */
2194        int r1a = sa.indexOf(v1);
2195        int r2a = sa.indexOf(v2);
2196
2197        int r2b = sb.indexOf(v2);
2198        int r1b = sb.indexOf(v1);
2199
2200        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
2201    }
2202
2203    static void indexOf_indexOf_pre_c_60(ArrayList sa, ArrayList sb, Object v1, Object
            v2)
2204    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2205                  sa..contents = sb..contents & sa..csize = sb..csize"
2206        ensures "True" */
2207    {
2208        /*: assume "~(True)" */
2209        int r1a = sa.indexOf(v1);
2210        int r2a = sa.indexOf(v2);
2211
2212        int r2b = sb.indexOf(v2);
2213        int r1b = sb.indexOf(v1);
2214
2215        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
2216    }
2217
2218    static void indexOf_indexOf_between_s_61(ArrayList sa, ArrayList sb, Object v1,
            Object v2)
2219    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2220                  sa..contents = sb..contents & sa..csize = sb..csize"
2221        ensures "True" */
2222    {
2223        int r1a = sa.indexOf(v1);
2224        /*: assume "True" */
2225        int r2a = sa.indexOf(v2);
```

```
2226
2227        int r2b = sb.indexOf(v2);
2228        int r1b = sb.indexOf(v1);
2229
2230        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
2231    }
2232
2233    static void indexOf_indexOf_between_c_61(ArrayList sa, ArrayList sb, Object v1,
            Object v2)
2234    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2235                  sa..contents = sb..contents & sa..csize = sb..csize"
2236        ensures "True" */
2237    {
2238        int r1a = sa.indexOf(v1);
2239        /*: assume "~(True)" */
2240        int r2a = sa.indexOf(v2);
2241
2242        int r2b = sb.indexOf(v2);
2243        int r1b = sb.indexOf(v1);
2244
2245        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
2246    }
2247
2248    static void indexOf_indexOf_post_s_62(ArrayList sa, ArrayList sb, Object v1, Object
            v2)
2249    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2250                  sa..contents = sb..contents & sa..csize = sb..csize"
2251        ensures "True" */
2252    {
2253        int r1a = sa.indexOf(v1);
2254        int r2a = sa.indexOf(v2);
2255        /*: assume "True" */
2256
2257        int r2b = sb.indexOf(v2);
2258        int r1b = sb.indexOf(v1);
2259
2260        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
2261    }
2262
2263    static void indexOf_indexOf_post_c_62(ArrayList sa, ArrayList sb, Object v1, Object
            v2)
2264    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2265                  sa..contents = sb..contents & sa..csize = sb..csize"
2266        ensures "True" */
2267    {
2268        int r1a = sa.indexOf(v1);
2269        int r2a = sa.indexOf(v2);
2270        /*: assume "~(True)" */
2271
2272        int r2b = sb.indexOf(v2);
2273        int r1b = sb.indexOf(v1);
2274
2275        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
2276    }
2277
2278    static void indexOf_lastIndexOf_pre_s_63(ArrayList sa, ArrayList sb, Object v1,
            Object v2)
2279    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2280                  sa..contents = sb..contents & sa..csize = sb..csize"
2281        ensures "True" */
2282    {
```

```
2283        /*: assume "True" */
2284        int r1a = sa.indexOf(v1);
2285        int r2a = sa.lastIndexOf(v2);
2286
2287        int r2b = sb.lastIndexOf(v2);
2288        int r1b = sb.indexOf(v1);
2289
2290        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
2291    }
2292
2293    static void indexOf_lastIndexOf_pre_c_63(ArrayList sa, ArrayList sb, Object v1,
            Object v2)
2294    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2295                  sa..contents = sb..contents & sa..csize = sb..csize"
2296        ensures "True" */
2297    {
2298        /*: assume "~(True)" */
2299        int r1a = sa.indexOf(v1);
2300        int r2a = sa.lastIndexOf(v2);
2301
2302        int r2b = sb.lastIndexOf(v2);
2303        int r1b = sb.indexOf(v1);
2304
2305        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
2306    }
2307
2308    static void indexOf_lastIndexOf_between_s_64(ArrayList sa, ArrayList sb, Object v1,
            Object v2)
2309    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2310                  sa..contents = sb..contents & sa..csize = sb..csize"
2311        ensures "True" */
2312    {
2313        int r1a = sa.indexOf(v1);
2314        /*: assume "True" */
2315        int r2a = sa.lastIndexOf(v2);
2316
2317        int r2b = sb.lastIndexOf(v2);
2318        int r1b = sb.indexOf(v1);
2319
2320        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
2321    }
2322
2323    static void indexOf_lastIndexOf_between_c_64(ArrayList sa, ArrayList sb, Object v1,
            Object v2)
2324    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2325                  sa..contents = sb..contents & sa..csize = sb..csize"
2326        ensures "True" */
2327    {
2328        int r1a = sa.indexOf(v1);
2329        /*: assume "~(True)" */
2330        int r2a = sa.lastIndexOf(v2);
2331
2332        int r2b = sb.lastIndexOf(v2);
2333        int r1b = sb.indexOf(v1);
2334
2335        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
2336    }
2337
2338    static void indexOf_lastIndexOf_post_s_65(ArrayList sa, ArrayList sb, Object v1,
            Object v2)
2339    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
```

```
2340                 sa..contents = sb..contents & sa..csize = sb..csize"
2341         ensures "True" */
2342     {
2343         int r1a = sa.indexOf(v1);
2344         int r2a = sa.lastIndexOf(v2);
2345         /*: assume "True" */
2346 
2347         int r2b = sb.lastIndexOf(v2);
2348         int r1b = sb.indexOf(v1);
2349 
2350         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
2351     }
2352 
2353     static void indexOf_lastIndexOf_post_c_65(ArrayList sa, ArrayList sb, Object v1,
            Object v2)
2354     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2355                 sa..contents = sb..contents & sa..csize = sb..csize"
2356         ensures "True" */
2357     {
2358         int r1a = sa.indexOf(v1);
2359         int r2a = sa.lastIndexOf(v2);
2360         /*: assume "~(True)" */
2361 
2362         int r2b = sb.lastIndexOf(v2);
2363         int r1b = sb.indexOf(v1);
2364 
2365         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
2366     }
2367 
2368     static void indexOf_remove_at_pre_s_66(ArrayList sa, ArrayList sb, Object v1, int
            i2)
2369     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2370                 sa..contents = sb..contents & sa..csize = sb..csize"
2371         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2372         ensures "True" */
2373     {
2374         /*: assume "~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
                (i, v1) : sa..contents & 0 <= i & i < i2) | (~(EX i. (i, v1) : sa..contents
                & 0 <= i & i < i2) & (i2, v1) : sa..contents & 0 <= i2 & i2 < sa..csize & i2
                < sa..csize - 1 & (i2 + 1, v1) : sa..contents & 0 <= i2 + 1 & i2 + 1 <
                sa..csize)" */
2375         int r1a = sa.indexOf(v1);
2376         /*: assume "0 <= i2 & i2 < sa..csize" */
2377         Object r2a = sa.remove_at(i2);
2378 
2379         Object r2b = sb.remove_at(i2);
2380         int r1b = sb.indexOf(v1);
2381 
2382         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
2383     }
2384 
2385     static void indexOf_remove_at_pre_c_66(ArrayList sa, ArrayList sb, Object v1, int
            i2)
2386     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2387                 sa..contents = sb..contents & sa..csize = sb..csize"
2388         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2389         ensures "True" */
2390     {
2391         /*: assume "~(~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
                (i, v1) : sa..contents & 0 <= i & i < i2) | (~(EX i. (i, v1) : sa..contents
                & 0 <= i & i < i2) & (i2, v1) : sa..contents & 0 <= i2 & i2 < sa..csize & i2
```

```
            < sa..csize - 1 & (i2 + 1, v1) : sa..contents & 0 <= i2 + 1 & i2 + 1 <
                sa..csize))" */
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.remove_at(i2);
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void indexOf_remove_at_between_s_67(ArrayList sa, ArrayList sb, Object v1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "r1a < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize - 1 &
            (i2 + 1, v1) : sa..contents & 0 <= i2 + 1 & i2 + 1 < sa..csize)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);

        Object r2b = sb.remove_at(i2);
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void indexOf_remove_at_between_c_67(ArrayList sa, ArrayList sb, Object v1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "~(r1a < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize - 1
            & (i2 + 1, v1) : sa..contents & 0 <= i2 + 1 & i2 + 1 < sa..csize))" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.remove_at(i2);
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void indexOf_remove_at_post_s_68(ArrayList sa, ArrayList sb, Object v1, int
        i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);
```

```
        /*: assume "r1a < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize & (i2,
            v1) : sa..contents & 0 <= i2 & i2 < sa..csize)" */

        Object r2b = sb.remove_at(i2);
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void indexOf_remove_at_post_c_68(ArrayList sa, ArrayList sb, Object v1, int
        i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);
        /*: assume "~(r1a < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize &
            (i2, v1) : sa..contents & 0 <= i2 & i2 < sa..csize))" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.remove_at(i2);
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void indexOf_remove_at_pre_s_69(ArrayList sa, ArrayList sb, Object v1, int
        i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
            (i, v1) : sa..contents & 0 <= i & i < i2) | (~(EX i. (i, v1) : sa..contents
            & 0 <= i & i < i2) & (i2, v1) : sa..contents & 0 <= i2 & i2 < sa..csize & i2
            < sa..csize - 1 & (i2 + 1, v1) : sa..contents & 0 <= i2 + 1 & i2 + 1 <
            sa..csize)" */
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        sb.remove_at(i2);
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void indexOf_remove_at_pre_c_69(ArrayList sa, ArrayList sb, Object v1, int
        i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~(~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
            (i, v1) : sa..contents & 0 <= i & i < i2) | (~(EX i. (i, v1) : sa..contents
            & 0 <= i & i < i2) & (i2, v1) : sa..contents & 0 <= i2 & i2 < sa..csize & i2
            < sa..csize - 1 & (i2 + 1, v1) : sa..contents & 0 <= i2 + 1 & i2 + 1 <
            sa..csize))" */
```

```
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void indexOf_remove_at_between_s_70(ArrayList sa, ArrayList sb, Object v1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "r1a < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize - 1 &
            (i2 + 1, v1) : sa..contents & 0 <= i2 + 1 & i2 + 1 < sa..csize)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        sb.remove_at(i2);
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void indexOf_remove_at_between_c_70(ArrayList sa, ArrayList sb, Object v1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "~(r1a < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize - 1
            & (i2 + 1, v1) : sa..contents & 0 <= i2 + 1 & i2 + 1 < sa..csize))" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void indexOf_remove_at_post_s_71(ArrayList sa, ArrayList sb, Object v1, int
        i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);
        /*: assume "r1a < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize & (i2,
            v1) : sa..contents & 0 <= i2 & i2 < sa..csize)" */

```

```
        sb.remove_at(i2);
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void indexOf_remove_at_post_c_71(ArrayList sa, ArrayList sb, Object v1, int
        i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);
        /*: assume "~(r1a < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize &
            (i2, v1) : sa..contents & 0 <= i2 & i2 < sa..csize))" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void indexOf_set_pre_s_72(ArrayList sa, ArrayList sb, Object v1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "(~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
            v2) | (EX i. (i, v1) : sa..contents & 0 <= i & i < i2) | (~(EX i. (i, v1) :
            sa..contents & 0 <= i & i < i2) & (i2, v1) : sa..contents & v1 = v2) | (~(EX
            i. (i, v1) : sa..contents & 0 <= i & i <= i2) & (EX i. (i, v1) :
            sa..contents & i2 < i & i < sa..csize) & v1 ~= v2)" */
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        Object r2b = sb.set(i2, v2);
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void indexOf_set_pre_c_72(ArrayList sa, ArrayList sb, Object v1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "~((~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
            v2) | (EX i. (i, v1) : sa..contents & 0 <= i & i < i2) | (~(EX i. (i, v1) :
            sa..contents & 0 <= i & i < i2) & (i2, v1) : sa..contents & v1 = v2) | (~(EX
            i. (i, v1) : sa..contents & 0 <= i & i <= i2) & (EX i. (i, v1) :
            sa..contents & i2 < i & i < sa..csize) & v1 ~= v2))" */
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);
```

```
2605
2606        /*: assume "0 <= i2 & i2 < sb..csize" */
2607        Object r2b = sb.set(i2, v2);
2608        int r1b = sb.indexOf(v1);
2609
2610        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
2611    }
2612
2613    static void indexOf_set_between_s_73(ArrayList sa, ArrayList sb, Object v1, int i2,
            Object v2)
2614    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2615                  sa..contents = sb..contents & sa..csize = sb..csize"
2616        modifies "sa..contents", "sb..contents"
2617        ensures "True" */
2618    {
2619        int r1a = sa.indexOf(v1);
2620        /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 = v2)
                | (r1a > i2 & v1 ~= v2)" */
2621        /*: assume "0 <= i2 & i2 < sa..csize" */
2622        Object r2a = sa.set(i2, v2);
2623
2624        Object r2b = sb.set(i2, v2);
2625        int r1b = sb.indexOf(v1);
2626
2627        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
2628    }
2629
2630    static void indexOf_set_between_c_73(ArrayList sa, ArrayList sb, Object v1, int i2,
            Object v2)
2631    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2632                  sa..contents = sb..contents & sa..csize = sb..csize"
2633        modifies "sa..contents", "sb..contents"
2634        ensures "True" */
2635    {
2636        int r1a = sa.indexOf(v1);
2637        /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 =
                v2) | (r1a > i2 & v1 ~= v2))" */
2638        /*: assume "0 <= i2 & i2 < sa..csize" */
2639        Object r2a = sa.set(i2, v2);
2640
2641        /*: assume "0 <= i2 & i2 < sb..csize" */
2642        Object r2b = sb.set(i2, v2);
2643        int r1b = sb.indexOf(v1);
2644
2645        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
2646    }
2647
2648    static void indexOf_set_post_s_74(ArrayList sa, ArrayList sb, Object v1, int i2,
            Object v2)
2649    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2650                  sa..contents = sb..contents & sa..csize = sb..csize"
2651        modifies "sa..contents", "sb..contents"
2652        ensures "True" */
2653    {
2654        int r1a = sa.indexOf(v1);
2655        /*: assume "0 <= i2 & i2 < sa..csize" */
2656        Object r2a = sa.set(i2, v2);
2657        /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 = v2)
                | (r1a > i2 & v1 ~= v2)" */
2658
2659        Object r2b = sb.set(i2, v2);
2660        int r1b = sb.indexOf(v1);
```

```
2661
2662        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
2663    }
2664
2665    static void indexOf_set_post_c_74(ArrayList sa, ArrayList sb, Object v1, int i2,
            Object v2)
2666    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2667                  sa..contents = sb..contents & sa..csize = sb..csize"
2668        modifies "sa..contents", "sb..contents"
2669        ensures "True" */
2670    {
2671        int r1a = sa.indexOf(v1);
2672        /*: assume "0 <= i2 & i2 < sa..csize" */
2673        Object r2a = sa.set(i2, v2);
2674        /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 =
                v2) | (r1a > i2 & v1 ~= v2))" */
2675
2676        /*: assume "0 <= i2 & i2 < sb..csize" */
2677        Object r2b = sb.set(i2, v2);
2678        int r1b = sb.indexOf(v1);
2679
2680        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
2681    }
2682
2683    static void indexOf_set_pre_s_75(ArrayList sa, ArrayList sb, Object v1, int i2,
            Object v2)
2684    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2685                  sa..contents = sb..contents & sa..csize = sb..csize"
2686        modifies "sa..contents", "sb..contents"
2687        ensures "True" */
2688    {
2689        /*: assume "(~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
                v2) | (EX i. (i, v1) : sa..contents & 0 <= i & i < i2) | (~(EX i. (i, v1) :
                sa..contents & 0 <= i & i < i2) & (i2, v1) : sa..contents & v1 = v2) | (~(EX
                i. (i, v1) : sa..contents & 0 <= i & i <= i2) & (EX i. (i, v1) :
                sa..contents & i2 < i & i < sa..csize) & v1 ~= v2)" */
2690        int r1a = sa.indexOf(v1);
2691        /*: assume "0 <= i2 & i2 < sa..csize" */
2692        sa.set(i2, v2);
2693
2694        sb.set(i2, v2);
2695        int r1b = sb.indexOf(v1);
2696
2697        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
2698    }
2699
2700    static void indexOf_set_pre_c_75(ArrayList sa, ArrayList sb, Object v1, int i2,
            Object v2)
2701    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2702                  sa..contents = sb..contents & sa..csize = sb..csize"
2703        modifies "sa..contents", "sb..contents"
2704        ensures "True" */
2705    {
2706        /*: assume "~((~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
                v2) | (EX i. (i, v1) : sa..contents & 0 <= i & i < i2) | (~(EX i. (i, v1) :
                sa..contents & 0 <= i & i < i2) & (i2, v1) : sa..contents & v1 = v2) | (~(EX
                i. (i, v1) : sa..contents & 0 <= i & i <= i2) & (EX i. (i, v1) :
                sa..contents & i2 < i & i < sa..csize) & v1 ~= v2))" */
2707        int r1a = sa.indexOf(v1);
2708        /*: assume "0 <= i2 & i2 < sa..csize" */
2709        sa.set(i2, v2);
2710
2711        /*: assume "0 <= i2 & i2 < sb..csize" */
```

```
        sb.set(i2, v2);
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)" */
    }

    static void indexOf_set_between_s_76(ArrayList sa, ArrayList sb, Object v1, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 = v2) | (r1a > i2 & v1 ~= v2)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        sb.set(i2, v2);
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void indexOf_set_between_c_76(ArrayList sa, ArrayList sb, Object v1, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 = v2) | (r1a > i2 & v1 ~= v2))" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.set(i2, v2);
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)" */
    }

    static void indexOf_set_post_s_77(ArrayList sa, ArrayList sb, Object v1, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);
        /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 = v2) | (r1a > i2 & v1 ~= v2)" */

        sb.set(i2, v2);
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }
```

```
2769
2770    static void indexOf_set_post_c_77(ArrayList sa, ArrayList sb, Object v1, int i2,
            Object v2)
2771    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2772                sa..contents = sb..contents & sa..csize = sb..csize"
2773        modifies "sa..contents", "sb..contents"
2774        ensures "True" */
2775    {
2776        int r1a = sa.indexOf(v1);
2777        /*: assume "0 <= i2 & i2 < sa..csize" */
2778        sa.set(i2, v2);
2779        /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 =
                v2) | (r1a > i2 & v1 ~= v2))" */
2780
2781        /*: assume "0 <= i2 & i2 < sb..csize" */
2782        sb.set(i2, v2);
2783        int r1b = sb.indexOf(v1);
2784
2785        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
2786    }
2787
2788    static void indexOf_size_pre_s_78(ArrayList sa, ArrayList sb, Object v1)
2789    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2790                sa..contents = sb..contents & sa..csize = sb..csize"
2791        ensures "True" */
2792    {
2793        /*: assume "True" */
2794        int r1a = sa.indexOf(v1);
2795        int r2a = sa.size();
2796
2797        int r2b = sb.size();
2798        int r1b = sb.indexOf(v1);
2799
2800        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
2801    }
2802
2803    static void indexOf_size_pre_c_78(ArrayList sa, ArrayList sb, Object v1)
2804    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2805                sa..contents = sb..contents & sa..csize = sb..csize"
2806        ensures "True" */
2807    {
2808        /*: assume "~(True)" */
2809        int r1a = sa.indexOf(v1);
2810        int r2a = sa.size();
2811
2812        int r2b = sb.size();
2813        int r1b = sb.indexOf(v1);
2814
2815        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
2816    }
2817
2818    static void indexOf_size_between_s_79(ArrayList sa, ArrayList sb, Object v1)
2819    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2820                sa..contents = sb..contents & sa..csize = sb..csize"
2821        ensures "True" */
2822    {
2823        int r1a = sa.indexOf(v1);
2824        /*: assume "True" */
2825        int r2a = sa.size();
2826
2827        int r2b = sb.size();
2828        int r1b = sb.indexOf(v1);
```

```
2829
2830        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
2831    }
2832
2833    static void indexOf_size_between_c_79(ArrayList sa, ArrayList sb, Object v1)
2834    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2835                  sa..contents = sb..contents & sa..csize = sb..csize"
2836        ensures "True" */
2837    {
2838        int r1a = sa.indexOf(v1);
2839        /*: assume "~(True)" */
2840        int r2a = sa.size();
2841
2842        int r2b = sb.size();
2843        int r1b = sb.indexOf(v1);
2844
2845        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
2846    }
2847
2848    static void indexOf_size_post_s_80(ArrayList sa, ArrayList sb, Object v1)
2849    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2850                  sa..contents = sb..contents & sa..csize = sb..csize"
2851        ensures "True" */
2852    {
2853        int r1a = sa.indexOf(v1);
2854        int r2a = sa.size();
2855        /*: assume "True" */
2856
2857        int r2b = sb.size();
2858        int r1b = sb.indexOf(v1);
2859
2860        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
2861    }
2862
2863    static void indexOf_size_post_c_80(ArrayList sa, ArrayList sb, Object v1)
2864    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2865                  sa..contents = sb..contents & sa..csize = sb..csize"
2866        ensures "True" */
2867    {
2868        int r1a = sa.indexOf(v1);
2869        int r2a = sa.size();
2870        /*: assume "~(True)" */
2871
2872        int r2b = sb.size();
2873        int r1b = sb.indexOf(v1);
2874
2875        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
2876    }
2877
2878    static void lastIndexOf_add_at_pre_s_81(ArrayList sa, ArrayList sb, Object v1, int
            i2, Object v2)
2879    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2880                  sa..contents = sb..contents & sa..csize = sb..csize"
2881        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
2882        ensures "True" */
2883    {
2884        /*: assume "(~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
                v2) | ((EX i. (i, v1) : sa..contents & 0 <= i & i < i2) & ~(EX i. (i, v1) :
                sa..contents & i2 <= i & i < sa..csize) & v1 ~= v2)" */
2885        int r1a = sa.lastIndexOf(v1);
```

```
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        sb.add_at(i2, v2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void lastIndexOf_add_at_pre_c_81(ArrayList sa, ArrayList sb, Object v1, int
        i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "~((~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
            v2) | ((EX i. (i, v1) : sa..contents & 0 <= i & i < i2) & ~(EX i. (i, v1) :
            sa..contents & i2 <= i & i < sa..csize) & v1 ~= v2))" */
        int r1a = sa.lastIndexOf(v1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        /*: assume "0 <= i2 & i2 <= sb..csize" */
        sb.add_at(i2, v2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void lastIndexOf_add_at_between_s_82(ArrayList sa, ArrayList sb, Object v1,
        int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        int r1a = sa.lastIndexOf(v1);
        /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2)" */
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        sb.add_at(i2, v2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void lastIndexOf_add_at_between_c_82(ArrayList sa, ArrayList sb, Object v1,
        int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        int r1a = sa.lastIndexOf(v1);
        /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2))" */
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        /*: assume "0 <= i2 & i2 <= sb..csize" */
```

```
2942        sb.add_at(i2, v2);
2943        int r1b = sb.lastIndexOf(v1);
2944
2945        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
2946    }
2947
2948    static void lastIndexOf_add_at_post_s_83(ArrayList sa, ArrayList sb, Object v1, int
            i2, Object v2)
2949    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2950                sa..contents = sb..contents & sa..csize = sb..csize"
2951        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
2952        ensures "True" */
2953    {
2954        int r1a = sa.lastIndexOf(v1);
2955        /*: assume "0 <= i2 & i2 <= sa..csize" */
2956        sa.add_at(i2, v2);
2957        /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2)" */
2958
2959        sb.add_at(i2, v2);
2960        int r1b = sb.lastIndexOf(v1);
2961
2962        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
2963    }
2964
2965    static void lastIndexOf_add_at_post_c_83(ArrayList sa, ArrayList sb, Object v1, int
            i2, Object v2)
2966    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2967                sa..contents = sb..contents & sa..csize = sb..csize"
2968        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
2969        ensures "True" */
2970    {
2971        int r1a = sa.lastIndexOf(v1);
2972        /*: assume "0 <= i2 & i2 <= sa..csize" */
2973        sa.add_at(i2, v2);
2974        /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2))" */
2975
2976        /*: assume "0 <= i2 & i2 <= sb..csize" */
2977        sb.add_at(i2, v2);
2978        int r1b = sb.lastIndexOf(v1);
2979
2980        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
2981    }
2982
2983    static void lastIndexOf_get_pre_s_84(ArrayList sa, ArrayList sb, Object v1, int i2)
2984    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2985                sa..contents = sb..contents & sa..csize = sb..csize"
2986        ensures "True" */
2987    {
2988        /*: assume "True" */
2989        int r1a = sa.lastIndexOf(v1);
2990        /*: assume "0 <= i2 & i2 < sa..csize" */
2991        Object r2a = sa.get(i2);
2992
2993        Object r2b = sb.get(i2);
2994        int r1b = sb.lastIndexOf(v1);
2995
2996        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
2997    }
2998
2999    static void lastIndexOf_get_pre_c_84(ArrayList sa, ArrayList sb, Object v1, int i2)
```

```
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "~(True)" */
        int r1a = sa.lastIndexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.get(i2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void lastIndexOf_get_between_s_85(ArrayList sa, ArrayList sb, Object v1, int
        i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.lastIndexOf(v1);
        /*: assume "True" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        Object r2b = sb.get(i2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void lastIndexOf_get_between_c_85(ArrayList sa, ArrayList sb, Object v1, int
        i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.lastIndexOf(v1);
        /*: assume "~(True)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.get(i2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void lastIndexOf_get_post_s_86(ArrayList sa, ArrayList sb, Object v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.lastIndexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);
        /*: assume "True" */

        Object r2b = sb.get(i2);
```

```
3060        int r1b = sb.lastIndexOf(v1);
3061
3062        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
3063    }
3064
3065    static void lastIndexOf_get_post_c_86(ArrayList sa, ArrayList sb, Object v1, int i2)
3066    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3067                  sa..contents = sb..contents & sa..csize = sb..csize"
3068        ensures "True" */
3069    {
3070        int r1a = sa.lastIndexOf(v1);
3071        /*: assume "0 <= i2 & i2 < sa..csize" */
3072        Object r2a = sa.get(i2);
3073        /*: assume "~(True)" */
3074
3075        /*: assume "0 <= i2 & i2 < sb..csize" */
3076        Object r2b = sb.get(i2);
3077        int r1b = sb.lastIndexOf(v1);
3078
3079        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
3080    }
3081
3082    static void lastIndexOf_indexOf_pre_s_87(ArrayList sa, ArrayList sb, Object v1,
            Object v2)
3083    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3084                  sa..contents = sb..contents & sa..csize = sb..csize"
3085        ensures "True" */
3086    {
3087        /*: assume "True" */
3088        int r1a = sa.lastIndexOf(v1);
3089        int r2a = sa.indexOf(v2);
3090
3091        int r2b = sb.indexOf(v2);
3092        int r1b = sb.lastIndexOf(v1);
3093
3094        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
3095    }
3096
3097    static void lastIndexOf_indexOf_pre_c_87(ArrayList sa, ArrayList sb, Object v1,
            Object v2)
3098    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3099                  sa..contents = sb..contents & sa..csize = sb..csize"
3100        ensures "True" */
3101    {
3102        /*: assume "~(True)" */
3103        int r1a = sa.lastIndexOf(v1);
3104        int r2a = sa.indexOf(v2);
3105
3106        int r2b = sb.indexOf(v2);
3107        int r1b = sb.lastIndexOf(v1);
3108
3109        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
3110    }
3111
3112    static void lastIndexOf_indexOf_between_s_88(ArrayList sa, ArrayList sb, Object v1,
            Object v2)
3113    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3114                  sa..contents = sb..contents & sa..csize = sb..csize"
3115        ensures "True" */
3116    {
3117        int r1a = sa.lastIndexOf(v1);
```

```
3118            /*: assume "True" */
3119            int r2a = sa.indexOf(v2);
3120
3121            int r2b = sb.indexOf(v2);
3122            int r1b = sb.lastIndexOf(v1);
3123
3124            /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                    sb..csize" */
3125        }
3126
3127        static void lastIndexOf_indexOf_between_c_88(ArrayList sa, ArrayList sb, Object v1,
                Object v2)
3128        /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3129                      sa..contents = sb..contents & sa..csize = sb..csize"
3130            ensures "True" */
3131        {
3132            int r1a = sa.lastIndexOf(v1);
3133            /*: assume "~(True)" */
3134            int r2a = sa.indexOf(v2);
3135
3136            int r2b = sb.indexOf(v2);
3137            int r1b = sb.lastIndexOf(v1);
3138
3139            /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                    sb..csize)" */
3140        }
3141
3142        static void lastIndexOf_indexOf_post_s_89(ArrayList sa, ArrayList sb, Object v1,
                Object v2)
3143        /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3144                      sa..contents = sb..contents & sa..csize = sb..csize"
3145            ensures "True" */
3146        {
3147            int r1a = sa.lastIndexOf(v1);
3148            int r2a = sa.indexOf(v2);
3149            /*: assume "True" */
3150
3151            int r2b = sb.indexOf(v2);
3152            int r1b = sb.lastIndexOf(v1);
3153
3154            /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                    sb..csize" */
3155        }
3156
3157        static void lastIndexOf_indexOf_post_c_89(ArrayList sa, ArrayList sb, Object v1,
                Object v2)
3158        /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3159                      sa..contents = sb..contents & sa..csize = sb..csize"
3160            ensures "True" */
3161        {
3162            int r1a = sa.lastIndexOf(v1);
3163            int r2a = sa.indexOf(v2);
3164            /*: assume "~(True)" */
3165
3166            int r2b = sb.indexOf(v2);
3167            int r1b = sb.lastIndexOf(v1);
3168
3169            /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                    sb..csize)" */
3170        }
3171
3172        static void lastIndexOf_lastIndexOf_pre_s_90(ArrayList sa, ArrayList sb, Object v1,
                Object v2)
3173        /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3174                      sa..contents = sb..contents & sa..csize = sb..csize"
```

```
3175        ensures "True" */
3176    {
3177        /*: assume "True" */
3178        int r1a = sa.lastIndexOf(v1);
3179        int r2a = sa.lastIndexOf(v2);
3180
3181        int r2b = sb.lastIndexOf(v2);
3182        int r1b = sb.lastIndexOf(v1);
3183
3184        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
3185    }
3186
3187    static void lastIndexOf_lastIndexOf_pre_c_90(ArrayList sa, ArrayList sb, Object v1,
            Object v2)
3188    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3189                  sa..contents = sb..contents & sa..csize = sb..csize"
3190        ensures "True" */
3191    {
3192        /*: assume "~(True)" */
3193        int r1a = sa.lastIndexOf(v1);
3194        int r2a = sa.lastIndexOf(v2);
3195
3196        int r2b = sb.lastIndexOf(v2);
3197        int r1b = sb.lastIndexOf(v1);
3198
3199        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
3200    }
3201
3202    static void lastIndexOf_lastIndexOf_between_s_91(ArrayList sa, ArrayList sb, Object
            v1, Object v2)
3203    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3204                  sa..contents = sb..contents & sa..csize = sb..csize"
3205        ensures "True" */
3206    {
3207        int r1a = sa.lastIndexOf(v1);
3208        /*: assume "True" */
3209        int r2a = sa.lastIndexOf(v2);
3210
3211        int r2b = sb.lastIndexOf(v2);
3212        int r1b = sb.lastIndexOf(v1);
3213
3214        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
3215    }
3216
3217    static void lastIndexOf_lastIndexOf_between_c_91(ArrayList sa, ArrayList sb, Object
            v1, Object v2)
3218    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3219                  sa..contents = sb..contents & sa..csize = sb..csize"
3220        ensures "True" */
3221    {
3222        int r1a = sa.lastIndexOf(v1);
3223        /*: assume "~(True)" */
3224        int r2a = sa.lastIndexOf(v2);
3225
3226        int r2b = sb.lastIndexOf(v2);
3227        int r1b = sb.lastIndexOf(v1);
3228
3229        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
3230    }
3231
```

```
3232    static void lastIndexOf_lastIndexOf_post_s_92(ArrayList sa, ArrayList sb, Object
            v1, Object v2)
3233    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3234                sa..contents = sb..contents & sa..csize = sb..csize"
3235        ensures "True" */
3236    {
3237        int r1a = sa.lastIndexOf(v1);
3238        int r2a = sa.lastIndexOf(v2);
3239        /*: assume "True" */
3240
3241        int r2b = sb.lastIndexOf(v2);
3242        int r1b = sb.lastIndexOf(v1);
3243
3244        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
3245    }
3246
3247    static void lastIndexOf_lastIndexOf_post_c_92(ArrayList sa, ArrayList sb, Object
            v1, Object v2)
3248    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3249                sa..contents = sb..contents & sa..csize = sb..csize"
3250        ensures "True" */
3251    {
3252        int r1a = sa.lastIndexOf(v1);
3253        int r2a = sa.lastIndexOf(v2);
3254        /*: assume "~(True)" */
3255
3256        int r2b = sb.lastIndexOf(v2);
3257        int r1b = sb.lastIndexOf(v1);
3258
3259        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
3260    }
3261
3262    static void lastIndexOf_remove_at_pre_s_93(ArrayList sa, ArrayList sb, Object v1,
            int i2)
3263    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3264                sa..contents = sb..contents & sa..csize = sb..csize"
3265        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3266        ensures "True" */
3267    {
3268        /*: assume "~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) | ((EX i.
                (i, v1) : sa..contents & 0 <= i & i < i2) & ~(EX i. (i, v1) : sa..contents &
                i2 <= i & i < sa..csize))" */
3269        int r1a = sa.lastIndexOf(v1);
3270        /*: assume "0 <= i2 & i2 < sa..csize" */
3271        Object r2a = sa.remove_at(i2);
3272
3273        Object r2b = sb.remove_at(i2);
3274        int r1b = sb.lastIndexOf(v1);
3275
3276        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
3277    }
3278
3279    static void lastIndexOf_remove_at_pre_c_93(ArrayList sa, ArrayList sb, Object v1,
            int i2)
3280    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3281                sa..contents = sb..contents & sa..csize = sb..csize"
3282        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3283        ensures "True" */
3284    {
3285        /*: assume "~(~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) | ((EX
                i. (i, v1) : sa..contents & 0 <= i & i < i2) & ~(EX i. (i, v1) :
                sa..contents & i2 <= i & i < sa..csize)))" */
```

```
3286        int r1a = sa.lastIndexOf(v1);
3287        /*: assume "0 <= i2 & i2 < sa..csize" */
3288        Object r2a = sa.remove_at(i2);
3289
3290        /*: assume "0 <= i2 & i2 < sb..csize" */
3291        Object r2b = sb.remove_at(i2);
3292        int r1b = sb.lastIndexOf(v1);
3293
3294        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
3295    }
3296
3297    static void lastIndexOf_remove_at_between_s_94(ArrayList sa, ArrayList sb, Object
            v1, int i2)
3298    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3299                  sa..contents = sb..contents & sa..csize = sb..csize"
3300        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3301        ensures "True" */
3302    {
3303        int r1a = sa.lastIndexOf(v1);
3304        /*: assume "r1a < 0 | (0 <= r1a & r1a < i2)" */
3305        /*: assume "0 <= i2 & i2 < sa..csize" */
3306        Object r2a = sa.remove_at(i2);
3307
3308        Object r2b = sb.remove_at(i2);
3309        int r1b = sb.lastIndexOf(v1);
3310
3311        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
3312    }
3313
3314    static void lastIndexOf_remove_at_between_c_94(ArrayList sa, ArrayList sb, Object
            v1, int i2)
3315    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3316                  sa..contents = sb..contents & sa..csize = sb..csize"
3317        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3318        ensures "True" */
3319    {
3320        int r1a = sa.lastIndexOf(v1);
3321        /*: assume "~(r1a < 0 | (0 <= r1a & r1a < i2))" */
3322        /*: assume "0 <= i2 & i2 < sa..csize" */
3323        Object r2a = sa.remove_at(i2);
3324
3325        /*: assume "0 <= i2 & i2 < sb..csize" */
3326        Object r2b = sb.remove_at(i2);
3327        int r1b = sb.lastIndexOf(v1);
3328
3329        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
3330    }
3331
3332    static void lastIndexOf_remove_at_post_s_95(ArrayList sa, ArrayList sb, Object v1,
            int i2)
3333    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3334                  sa..contents = sb..contents & sa..csize = sb..csize"
3335        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3336        ensures "True" */
3337    {
3338        int r1a = sa.lastIndexOf(v1);
3339        /*: assume "0 <= i2 & i2 < sa..csize" */
3340        Object r2a = sa.remove_at(i2);
3341        /*: assume "r1a < 0 | (0 <= r1a & r1a < i2)" */
3342
3343        Object r2b = sb.remove_at(i2);
3344        int r1b = sb.lastIndexOf(v1);
```

```
3345
3346        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
3347    }
3348
3349    static void lastIndexOf_remove_at_post_c_95(ArrayList sa, ArrayList sb, Object v1,
            int i2)
3350    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3351                  sa..contents = sb..contents & sa..csize = sb..csize"
3352        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3353        ensures "True" */
3354    {
3355        int r1a = sa.lastIndexOf(v1);
3356        /*: assume "0 <= i2 & i2 < sa..csize" */
3357        Object r2a = sa.remove_at(i2);
3358        /*: assume "~(r1a < 0 | (0 <= r1a & r1a < i2))" */
3359
3360        /*: assume "0 <= i2 & i2 < sb..csize" */
3361        Object r2b = sb.remove_at(i2);
3362        int r1b = sb.lastIndexOf(v1);
3363
3364        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
3365    }
3366
3367    static void lastIndexOf_remove_at_pre_s_96(ArrayList sa, ArrayList sb, Object v1,
            int i2)
3368    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3369                  sa..contents = sb..contents & sa..csize = sb..csize"
3370        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3371        ensures "True" */
3372    {
3373        /*: assume "~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) | ((EX i.
                (i, v1) : sa..contents & 0 <= i & i < i2) & ~(EX i. (i, v1) : sa..contents &
                i2 <= i & i < sa..csize))" */
3374        int r1a = sa.lastIndexOf(v1);
3375        /*: assume "0 <= i2 & i2 < sa..csize" */
3376        sa.remove_at(i2);
3377
3378        sb.remove_at(i2);
3379        int r1b = sb.lastIndexOf(v1);
3380
3381        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
3382    }
3383
3384    static void lastIndexOf_remove_at_pre_c_96(ArrayList sa, ArrayList sb, Object v1,
            int i2)
3385    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3386                  sa..contents = sb..contents & sa..csize = sb..csize"
3387        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3388        ensures "True" */
3389    {
3390        /*: assume "~(~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) | ((EX
                i. (i, v1) : sa..contents & 0 <= i & i < i2) & ~(EX i. (i, v1) :
                sa..contents & i2 <= i & i < sa..csize)))" */
3391        int r1a = sa.lastIndexOf(v1);
3392        /*: assume "0 <= i2 & i2 < sa..csize" */
3393        sa.remove_at(i2);
3394
3395        /*: assume "0 <= i2 & i2 < sb..csize" */
3396        sb.remove_at(i2);
3397        int r1b = sb.lastIndexOf(v1);
3398
3399        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
```

```
3400    }
3401
3402    static void lastIndexOf_remove_at_between_s_97(ArrayList sa, ArrayList sb, Object
            v1, int i2)
3403    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3404                sa..contents = sb..contents & sa..csize = sb..csize"
3405        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3406        ensures "True" */
3407    {
3408        int r1a = sa.lastIndexOf(v1);
3409        /*: assume "r1a < 0 | (0 <= r1a & r1a < i2)" */
3410        /*: assume "0 <= i2 & i2 < sa..csize" */
3411        sa.remove_at(i2);
3412
3413        sb.remove_at(i2);
3414        int r1b = sb.lastIndexOf(v1);
3415
3416        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
3417    }
3418
3419    static void lastIndexOf_remove_at_between_c_97(ArrayList sa, ArrayList sb, Object
            v1, int i2)
3420    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3421                sa..contents = sb..contents & sa..csize = sb..csize"
3422        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3423        ensures "True" */
3424    {
3425        int r1a = sa.lastIndexOf(v1);
3426        /*: assume "~(r1a < 0 | (0 <= r1a & r1a < i2))" */
3427        /*: assume "0 <= i2 & i2 < sa..csize" */
3428        sa.remove_at(i2);
3429
3430        /*: assume "0 <= i2 & i2 < sb..csize" */
3431        sb.remove_at(i2);
3432        int r1b = sb.lastIndexOf(v1);
3433
3434        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
3435    }
3436
3437    static void lastIndexOf_remove_at_post_s_98(ArrayList sa, ArrayList sb, Object v1,
            int i2)
3438    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3439                sa..contents = sb..contents & sa..csize = sb..csize"
3440        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3441        ensures "True" */
3442    {
3443        int r1a = sa.lastIndexOf(v1);
3444        /*: assume "0 <= i2 & i2 < sa..csize" */
3445        sa.remove_at(i2);
3446        /*: assume "r1a < 0 | (0 <= r1a & r1a < i2)" */
3447
3448        sb.remove_at(i2);
3449        int r1b = sb.lastIndexOf(v1);
3450
3451        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
3452    }
3453
3454    static void lastIndexOf_remove_at_post_c_98(ArrayList sa, ArrayList sb, Object v1,
            int i2)
3455    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3456                sa..contents = sb..contents & sa..csize = sb..csize"
3457        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3458        ensures "True" */
3459    {
```

```
        int r1a = sa.lastIndexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);
        /*: assume "~(r1a < 0 | (0 <= r1a & r1a < i2))" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void lastIndexOf_set_pre_s_99(ArrayList sa, ArrayList sb, Object v1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "(~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
            v2) | ((EX i. (i, v1) : sa..contents & 0 <= i & i < i2) & ~(EX i. (i, v1) :
            sa..contents & i2 <= i & i < sa..csize) & v1 ~= v2) | ((i2, v1) :
            sa..contents & 0 <= i2 & i2 < sa..csize & ~(EX i. (i, v1) : sa..contents &
            i2 < i & i < sa..csize) & v1 = v2) | (EX i. (i, v1) : sa..contents & i2 < i
            & i < sa..csize)" */
        int r1a = sa.lastIndexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        Object r2b = sb.set(i2, v2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void lastIndexOf_set_pre_c_99(ArrayList sa, ArrayList sb, Object v1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "~((~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
            v2) | ((EX i. (i, v1) : sa..contents & 0 <= i & i < i2) & ~(EX i. (i, v1) :
            sa..contents & i2 <= i & i < sa..csize) & v1 ~= v2) | ((i2, v1) :
            sa..contents & 0 <= i2 & i2 < sa..csize & ~(EX i. (i, v1) : sa..contents &
            i2 < i & i < sa..csize) & v1 = v2) | (EX i. (i, v1) : sa..contents & i2 < i
            & i < sa..csize))" */
        int r1a = sa.lastIndexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.set(i2, v2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void lastIndexOf_set_between_s_100(ArrayList sa, ArrayList sb, Object v1,
        int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
```

```
3509                 sa..contents = sb..contents & sa..csize = sb..csize"
3510         modifies "sa..contents", "sb..contents"
3511         ensures "True" */
3512     {
3513         int r1a = sa.lastIndexOf(v1);
3514         /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2) | (r1a = i2
                & v1 = v2) | r1a > i2" */
3515         /*: assume "0 <= i2 & i2 < sa..csize" */
3516         Object r2a = sa.set(i2, v2);
3517 
3518         Object r2b = sb.set(i2, v2);
3519         int r1b = sb.lastIndexOf(v1);
3520 
3521         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
3522     }
3523 
3524     static void lastIndexOf_set_between_c_100(ArrayList sa, ArrayList sb, Object v1,
            int i2, Object v2)
3525     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3526                 sa..contents = sb..contents & sa..csize = sb..csize"
3527         modifies "sa..contents", "sb..contents"
3528         ensures "True" */
3529     {
3530         int r1a = sa.lastIndexOf(v1);
3531         /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2) | (r1a =
                i2 & v1 = v2) | r1a > i2)" */
3532         /*: assume "0 <= i2 & i2 < sa..csize" */
3533         Object r2a = sa.set(i2, v2);
3534 
3535         /*: assume "0 <= i2 & i2 < sb..csize" */
3536         Object r2b = sb.set(i2, v2);
3537         int r1b = sb.lastIndexOf(v1);
3538 
3539         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
3540     }
3541 
3542     static void lastIndexOf_set_post_s_101(ArrayList sa, ArrayList sb, Object v1, int
            i2, Object v2)
3543     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3544                 sa..contents = sb..contents & sa..csize = sb..csize"
3545         modifies "sa..contents", "sb..contents"
3546         ensures "True" */
3547     {
3548         int r1a = sa.lastIndexOf(v1);
3549         /*: assume "0 <= i2 & i2 < sa..csize" */
3550         Object r2a = sa.set(i2, v2);
3551         /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2) | (r1a = i2
                & v1 = v2) | r1a > i2" */
3552 
3553         Object r2b = sb.set(i2, v2);
3554         int r1b = sb.lastIndexOf(v1);
3555 
3556         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
3557     }
3558 
3559     static void lastIndexOf_set_post_c_101(ArrayList sa, ArrayList sb, Object v1, int
            i2, Object v2)
3560     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3561                 sa..contents = sb..contents & sa..csize = sb..csize"
3562         modifies "sa..contents", "sb..contents"
3563         ensures "True" */
3564     {
```

```
        int r1a = sa.lastIndexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);
        /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2) | (r1a =
            i2 & v1 = v2) | r1a > i2)" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.set(i2, v2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void lastIndexOf_set_pre_s_102(ArrayList sa, ArrayList sb, Object v1, int
        i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "(~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
            v2) | ((EX i. (i, v1) : sa..contents & 0 <= i & i < i2) & ~(EX i. (i, v1) :
            sa..contents & i2 <= i & i < sa..csize) & v1 ~= v2) | ((i2, v1) :
            sa..contents & 0 <= i2 & i2 < sa..csize & ~(EX i. (i, v1) : sa..contents &
            i2 < i & i < sa..csize) & v1 = v2) | (EX i. (i, v1) : sa..contents & i2 < i
            & i < sa..csize)" */
        int r1a = sa.lastIndexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        sb.set(i2, v2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void lastIndexOf_set_pre_c_102(ArrayList sa, ArrayList sb, Object v1, int
        i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "~((~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
            v2) | ((EX i. (i, v1) : sa..contents & 0 <= i & i < i2) & ~(EX i. (i, v1) :
            sa..contents & i2 <= i & i < sa..csize) & v1 ~= v2) | ((i2, v1) :
            sa..contents & 0 <= i2 & i2 < sa..csize & ~(EX i. (i, v1) : sa..contents &
            i2 < i & i < sa..csize) & v1 = v2) | (EX i. (i, v1) : sa..contents & i2 < i
            & i < sa..csize))" */
        int r1a = sa.lastIndexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.set(i2, v2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void lastIndexOf_set_between_s_103(ArrayList sa, ArrayList sb, Object v1,
        int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
```

```
3614                 sa..contents = sb..contents & sa..csize = sb..csize"
3615         modifies "sa..contents", "sb..contents"
3616         ensures "True" */
3617     {
3618         int r1a = sa.lastIndexOf(v1);
3619         /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2) | (r1a = i2
                & v1 = v2) | r1a > i2" */
3620         /*: assume "0 <= i2 & i2 < sa..csize" */
3621         sa.set(i2, v2);
3622 
3623         sb.set(i2, v2);
3624         int r1b = sb.lastIndexOf(v1);
3625 
3626         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
3627     }
3628 
3629     static void lastIndexOf_set_between_c_103(ArrayList sa, ArrayList sb, Object v1,
            int i2, Object v2)
3630     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3631                 sa..contents = sb..contents & sa..csize = sb..csize"
3632         modifies "sa..contents", "sb..contents"
3633         ensures "True" */
3634     {
3635         int r1a = sa.lastIndexOf(v1);
3636         /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2) | (r1a =
                i2 & v1 = v2) | r1a > i2)" */
3637         /*: assume "0 <= i2 & i2 < sa..csize" */
3638         sa.set(i2, v2);
3639 
3640         /*: assume "0 <= i2 & i2 < sb..csize" */
3641         sb.set(i2, v2);
3642         int r1b = sb.lastIndexOf(v1);
3643 
3644         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
3645     }
3646 
3647     static void lastIndexOf_set_post_s_104(ArrayList sa, ArrayList sb, Object v1, int
            i2, Object v2)
3648     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3649                 sa..contents = sb..contents & sa..csize = sb..csize"
3650         modifies "sa..contents", "sb..contents"
3651         ensures "True" */
3652     {
3653         int r1a = sa.lastIndexOf(v1);
3654         /*: assume "0 <= i2 & i2 < sa..csize" */
3655         sa.set(i2, v2);
3656         /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2) | (r1a = i2
                & v1 = v2) | r1a > i2" */
3657 
3658         sb.set(i2, v2);
3659         int r1b = sb.lastIndexOf(v1);
3660 
3661         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
3662     }
3663 
3664     static void lastIndexOf_set_post_c_104(ArrayList sa, ArrayList sb, Object v1, int
            i2, Object v2)
3665     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3666                 sa..contents = sb..contents & sa..csize = sb..csize"
3667         modifies "sa..contents", "sb..contents"
3668         ensures "True" */
3669     {
3670         int r1a = sa.lastIndexOf(v1);
3671         /*: assume "0 <= i2 & i2 < sa..csize" */
```

```
        sa.set(i2, v2);
        /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2) | (r1a =
            i2 & v1 = v2) | r1a > i2)" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.set(i2, v2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void lastIndexOf_size_pre_s_105(ArrayList sa, ArrayList sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "True" */
        int r1a = sa.lastIndexOf(v1);
        int r2a = sa.size();

        int r2b = sb.size();
        int r1b = sb.lastIndexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void lastIndexOf_size_pre_c_105(ArrayList sa, ArrayList sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "~(True)" */
        int r1a = sa.lastIndexOf(v1);
        int r2a = sa.size();

        int r2b = sb.size();
        int r1b = sb.lastIndexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void lastIndexOf_size_between_s_106(ArrayList sa, ArrayList sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.lastIndexOf(v1);
        /*: assume "True" */
        int r2a = sa.size();

        int r2b = sb.size();
        int r1b = sb.lastIndexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void lastIndexOf_size_between_c_106(ArrayList sa, ArrayList sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
```

```
3732        int r1a = sa.lastIndexOf(v1);
3733        /*: assume "~(True)" */
3734        int r2a = sa.size();
3735
3736        int r2b = sb.size();
3737        int r1b = sb.lastIndexOf(v1);
3738
3739        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
3740    }
3741
3742    static void lastIndexOf_size_post_s_107(ArrayList sa, ArrayList sb, Object v1)
3743    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3744                  sa..contents = sb..contents & sa..csize = sb..csize"
3745        ensures "True" */
3746    {
3747        int r1a = sa.lastIndexOf(v1);
3748        int r2a = sa.size();
3749        /*: assume "True" */
3750
3751        int r2b = sb.size();
3752        int r1b = sb.lastIndexOf(v1);
3753
3754        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
3755    }
3756
3757    static void lastIndexOf_size_post_c_107(ArrayList sa, ArrayList sb, Object v1)
3758    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3759                  sa..contents = sb..contents & sa..csize = sb..csize"
3760        ensures "True" */
3761    {
3762        int r1a = sa.lastIndexOf(v1);
3763        int r2a = sa.size();
3764        /*: assume "~(True)" */
3765
3766        int r2b = sb.size();
3767        int r1b = sb.lastIndexOf(v1);
3768
3769        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
3770    }
3771
3772    static void remove_at_add_at_pre_s_108(ArrayList sa, ArrayList sb, int i1, int i2,
            Object v2)
3773    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3774                  sa..contents = sb..contents & sa..csize = sb..csize"
3775        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
3776        ensures "True" */
3777    {
3778        /*: assume "(i1 < i2 & (i2, v2) : sa..contents & 0 <= i2 & i2 < sa..csize) |
                (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize) | (i1 > i2 &
                (ALL v. ((i1 - 1, v) : sa..contents) = ((i1, v) : sa..contents)) & 0 <= i1 -
                1 & i1 - 1 < sa..csize & 0 <= i1 & i1 < sa..csize)" */
3779        /*: assume "0 <= i1 & i1 < sa..csize" */
3780        Object r1a = sa.remove_at(i1);
3781        /*: assume "0 <= i2 & i2 <= sa..csize" */
3782        sa.add_at(i2, v2);
3783
3784        sb.add_at(i2, v2);
3785        Object r1b = sb.remove_at(i1);
3786
3787        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
3788    }
```

```
3789
3790    static void remove_at_add_at_pre_c_108(ArrayList sa, ArrayList sb, int i1, int i2,
            Object v2)
3791    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3792                 sa..contents = sb..contents & sa..csize = sb..csize"
3793        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
3794        ensures "True" */
3795    {
3796        /*: assume "~((i1 < i2 & (i2, v2) : sa..contents & 0 <= i2 & i2 < sa..csize) |
                (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize) | (i1 > i2 &
                (ALL v. ((i1 - 1, v) : sa..contents) = ((i1, v) : sa..contents)) & 0 <= i1 -
                1 & i1 - 1 < sa..csize & 0 <= i1 & i1 < sa..csize))" */
3797        /*: assume "0 <= i1 & i1 < sa..csize" */
3798        Object r1a = sa.remove_at(i1);
3799        /*: assume "0 <= i2 & i2 <= sa..csize" */
3800        sa.add_at(i2, v2);
3801
3802        /*: assume "0 <= i2 & i2 <= sb..csize" */
3803        sb.add_at(i2, v2);
3804        /*: assume "0 <= i1 & i1 < sb..csize" */
3805        Object r1b = sb.remove_at(i1);
3806
3807        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
3808    }
3809
3810    static void remove_at_add_at_between_s_109(ArrayList sa, ArrayList sb, int i1, int
            i2, Object v2)
3811    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3812                 sa..contents = sb..contents & sa..csize = sb..csize"
3813        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
3814        ensures "True" */
3815    {
3816        /*: assume "0 <= i1 & i1 < sa..csize" */
3817        Object r1a = sa.remove_at(i1);
3818        /*: assume "(i1 < i2 & (i2 - 1, v2) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
                sa..csize) | (i1 = i2 & r1a = v2) | (i1 > i2 & (i1 - 1, r1a) : sa..contents
                & 0 <= i1 - 1 & i1 - 1 < sa..csize)" */
3819        /*: assume "0 <= i2 & i2 <= sa..csize" */
3820        sa.add_at(i2, v2);
3821
3822        sb.add_at(i2, v2);
3823        Object r1b = sb.remove_at(i1);
3824
3825        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
3826    }
3827
3828    static void remove_at_add_at_between_c_109(ArrayList sa, ArrayList sb, int i1, int
            i2, Object v2)
3829    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3830                 sa..contents = sb..contents & sa..csize = sb..csize"
3831        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
3832        ensures "True" */
3833    {
3834        /*: assume "0 <= i1 & i1 < sa..csize" */
3835        Object r1a = sa.remove_at(i1);
3836        /*: assume "~((i1 < i2 & (i2 - 1, v2) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
                sa..csize) | (i1 = i2 & r1a = v2) | (i1 > i2 & (i1 - 1, r1a) : sa..contents
                & 0 <= i1 - 1 & i1 - 1 < sa..csize))" */
3837        /*: assume "0 <= i2 & i2 <= sa..csize" */
3838        sa.add_at(i2, v2);
3839
```

```
        /*: assume "0 <= i2 & i2 <= sb..csize" */
        sb.add_at(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.remove_at(i1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void remove_at_add_at_post_s_110(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);
        /*: assume "(i1 < i2 & (i2 - 1, v2) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
            sa..csize) | (i1 = i2 & r1a = v2) | (i1 > i2 & (i1, r1a) : sa..contents & 0
            <= i1 & i1 < sa..csize)" */

        sb.add_at(i2, v2);
        Object r1b = sb.remove_at(i1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_add_at_post_c_110(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);
        /*: assume "~((i1 < i2 & (i2 - 1, v2) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
            sa..csize) | (i1 = i2 & r1a = v2) | (i1 > i2 & (i1, r1a) : sa..contents & 0
            <= i1 & i1 < sa..csize))" */

        /*: assume "0 <= i2 & i2 <= sb..csize" */
        sb.add_at(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.remove_at(i1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void remove_at_get_pre_s_111(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :
```

```
            sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | i1 > i2" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        Object r2b = sb.get(i2);
        Object r1b = sb.remove_at(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void remove_at_get_pre_c_111(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | i1 > i2)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.get(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.remove_at(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void remove_at_get_between_s_112(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
            sa..csize) | (i1 = i2 & (i2, r1a) : sa..contents) | i1 > i2" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        Object r2b = sb.get(i2);
        Object r1b = sb.remove_at(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void remove_at_get_between_c_112(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
```

```
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
            sa..csize) | (i1 = i2 & (i2, r1a) : sa..contents) | i1 > i2)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.get(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.remove_at(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void remove_at_get_post_s_113(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);
        /*: assume "(i1 < i2 & (i2 - 1, r2a) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
            sa..csize) | (i1 = i2 & r1a = r2a) | i1 > i2" */

        Object r2b = sb.get(i2);
        Object r1b = sb.remove_at(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void remove_at_get_post_c_113(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);
        /*: assume "~((i1 < i2 & (i2 - 1, r2a) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
            sa..csize) | (i1 = i2 & r1a = r2a) | i1 > i2)" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.get(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.remove_at(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void remove_at_indexOf_pre_s_114(ArrayList sa, ArrayList sb, int i1, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
```

```
4004        ensures "True" */
4005    {
4006        /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
                (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) : sa..contents
                & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & i1
                < sa..csize - 1 & (i1 + 1, v2) : sa..contents & 0 <= i1 + 1 & i1 + 1 <
                sa..csize)" */
4007        /*: assume "0 <= i1 & i1 < sa..csize" */
4008        Object r1a = sa.remove_at(i1);
4009        int r2a = sa.indexOf(v2);
4010
4011        int r2b = sb.indexOf(v2);
4012        Object r1b = sb.remove_at(i1);
4013
4014        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
4015    }
4016
4017    static void remove_at_indexOf_pre_c_114(ArrayList sa, ArrayList sb, int i1, Object
            v2)
4018    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4019                sa..contents = sb..contents & sa..csize = sb..csize"
4020        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4021        ensures "True" */
4022    {
4023        /*: assume "~(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
                (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) : sa..contents
                & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & i1
                < sa..csize - 1 & (i1 + 1, v2) : sa..contents & 0 <= i1 + 1 & i1 + 1 <
                sa..csize))" */
4024        /*: assume "0 <= i1 & i1 < sa..csize" */
4025        Object r1a = sa.remove_at(i1);
4026        int r2a = sa.indexOf(v2);
4027
4028        int r2b = sb.indexOf(v2);
4029        /*: assume "0 <= i1 & i1 < sb..csize" */
4030        Object r1b = sb.remove_at(i1);
4031
4032        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
4033    }
4034
4035    static void remove_at_indexOf_between_s_115(ArrayList sa, ArrayList sb, int i1,
            Object v2)
4036    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4037                sa..contents = sb..contents & sa..csize = sb..csize"
4038        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4039        ensures "True" */
4040    {
4041        /*: assume "0 <= i1 & i1 < sa..csize" */
4042        Object r1a = sa.remove_at(i1);
4043        /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & r1a ~=
                v2) | (EX i. (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) :
                sa..contents & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 <
                sa..csize & r1a = v2 & i1 < sa..csize)" */
4044        int r2a = sa.indexOf(v2);
4045
4046        int r2b = sb.indexOf(v2);
4047        Object r1b = sb.remove_at(i1);
4048
4049        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
4050    }
4051
```

```
    static void remove_at_indexOf_between_c_115(ArrayList sa, ArrayList sb, int i1,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & r1a
            ~= v2) | (EX i. (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2)
            : sa..contents & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 <
            sa..csize & r1a = v2 & i1 < sa..csize))" */
        int r2a = sa.indexOf(v2);

        int r2b = sb.indexOf(v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.remove_at(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void remove_at_indexOf_post_s_116(ArrayList sa, ArrayList sb, int i1, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        int r2a = sa.indexOf(v2);
        /*: assume "(r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1) | (r2a = i1 & r1a =
            v2 & i1 < sa..csize)" */

        int r2b = sb.indexOf(v2);
        Object r1b = sb.remove_at(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void remove_at_indexOf_post_c_116(ArrayList sa, ArrayList sb, int i1, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        int r2a = sa.indexOf(v2);
        /*: assume "~((r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1) | (r2a = i1 & r1a =
            v2 & i1 < sa..csize))" */

        int r2b = sb.indexOf(v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.remove_at(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }
```

```
4105    static void remove_at_lastIndexOf_pre_s_117(ArrayList sa, ArrayList sb, int i1,
            Object v2)
4106    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4107                  sa..contents = sb..contents & sa..csize = sb..csize"
4108        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4109        ensures "True" */
4110    {
4111        /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | ((EX i.
                (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) : sa..contents &
                i1 <= i & i < sa..csize))" */
4112        /*: assume "0 <= i1 & i1 < sa..csize" */
4113        Object r1a = sa.remove_at(i1);
4114        int r2a = sa.lastIndexOf(v2);
4115
4116        int r2b = sb.lastIndexOf(v2);
4117        Object r1b = sb.remove_at(i1);
4118
4119        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
4120    }
4121
4122    static void remove_at_lastIndexOf_pre_c_117(ArrayList sa, ArrayList sb, int i1,
            Object v2)
4123    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4124                  sa..contents = sb..contents & sa..csize = sb..csize"
4125        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4126        ensures "True" */
4127    {
4128        /*: assume "~(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | ((EX
                i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
                sa..contents & i1 <= i & i < sa..csize)))" */
4129        /*: assume "0 <= i1 & i1 < sa..csize" */
4130        Object r1a = sa.remove_at(i1);
4131        int r2a = sa.lastIndexOf(v2);
4132
4133        int r2b = sb.lastIndexOf(v2);
4134        /*: assume "0 <= i1 & i1 < sb..csize" */
4135        Object r1b = sb.remove_at(i1);
4136
4137        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
4138    }
4139
4140    static void remove_at_lastIndexOf_between_s_118(ArrayList sa, ArrayList sb, int i1,
            Object v2)
4141    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4142                  sa..contents = sb..contents & sa..csize = sb..csize"
4143        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4144        ensures "True" */
4145    {
4146        /*: assume "0 <= i1 & i1 < sa..csize" */
4147        Object r1a = sa.remove_at(i1);
4148        /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & r1a ~=
                v2) | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
                sa..contents & i1 <= i & i < sa..csize) & r1a ~= v2)" */
4149        int r2a = sa.lastIndexOf(v2);
4150
4151        int r2b = sb.lastIndexOf(v2);
4152        Object r1b = sb.remove_at(i1);
4153
4154        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
4155    }
4156
```

```
4157    static void remove_at_lastIndexOf_between_c_118(ArrayList sa, ArrayList sb, int i1,
            Object v2)
4158    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4159                sa..contents = sb..contents & sa..csize = sb..csize"
4160        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4161        ensures "True" */
4162    {
4163        /*: assume "0 <= i1 & i1 < sa..csize" */
4164        Object r1a = sa.remove_at(i1);
4165        /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & r1a
                ~= v2) | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2)
                : sa..contents & i1 <= i & i < sa..csize) & r1a ~= v2))" */
4166        int r2a = sa.lastIndexOf(v2);
4167
4168        int r2b = sb.lastIndexOf(v2);
4169        /*: assume "0 <= i1 & i1 < sb..csize" */
4170        Object r1b = sb.remove_at(i1);
4171
4172        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
4173    }
4174
4175    static void remove_at_lastIndexOf_post_s_119(ArrayList sa, ArrayList sb, int i1,
            Object v2)
4176    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4177                sa..contents = sb..contents & sa..csize = sb..csize"
4178        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4179        ensures "True" */
4180    {
4181        /*: assume "0 <= i1 & i1 < sa..csize" */
4182        Object r1a = sa.remove_at(i1);
4183        int r2a = sa.lastIndexOf(v2);
4184        /*: assume "(r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1 & r1a ~= v2)" */
4185
4186        int r2b = sb.lastIndexOf(v2);
4187        Object r1b = sb.remove_at(i1);
4188
4189        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
4190    }
4191
4192    static void remove_at_lastIndexOf_post_c_119(ArrayList sa, ArrayList sb, int i1,
            Object v2)
4193    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4194                sa..contents = sb..contents & sa..csize = sb..csize"
4195        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4196        ensures "True" */
4197    {
4198        /*: assume "0 <= i1 & i1 < sa..csize" */
4199        Object r1a = sa.remove_at(i1);
4200        int r2a = sa.lastIndexOf(v2);
4201        /*: assume "~((r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1 & r1a ~= v2))" */
4202
4203        int r2b = sb.lastIndexOf(v2);
4204        /*: assume "0 <= i1 & i1 < sb..csize" */
4205        Object r1b = sb.remove_at(i1);
4206
4207        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
4208    }
4209
4210    static void remove_at_remove_at_pre_s_120(ArrayList sa, ArrayList sb, int i1, int
            i2)
4211    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4212                sa..contents = sb..contents & sa..csize = sb..csize"
```

```
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
            sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
            <= i1 + 1 & i1 + 1 < sa..csize)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);

        Object r2b = sb.remove_at(i2);
        Object r1b = sb.remove_at(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void remove_at_remove_at_pre_c_120(ArrayList sa, ArrayList sb, int i1, int
        i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
            sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
            <= i1 + 1 & i1 + 1 < sa..csize))" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.remove_at(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void remove_at_remove_at_between_s_121(ArrayList sa, ArrayList sb, int i1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
            sa..csize) | (i1 = i2 & (i2, r1a) : sa..contents & 0 <= i2 & i2 < sa..csize)
            | (sa..csize > i1 & i1 > i2 & (i1, r1a) : sa..contents & 0 <= i1 & i1 <
            sa..csize)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
```

```
4258        Object r2a = sa.remove_at(i2);
4259
4260        Object r2b = sb.remove_at(i2);
4261        Object r1b = sb.remove_at(i1);
4262
4263        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
4264    }
4265
4266    static void remove_at_remove_at_between_c_121(ArrayList sa, ArrayList sb, int i1,
            int i2)
4267    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4268                    sa..contents = sb..contents & sa..csize = sb..csize"
4269        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4270        ensures "True" */
4271    {
4272        /*: assume "0 <= i1 & i1 < sa..csize" */
4273        Object r1a = sa.remove_at(i1);
4274        /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
                sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
                sa..csize) | (i1 = i2 & (i2, r1a) : sa..contents & 0 <= i2 & i2 < sa..csize)
                | (sa..csize > i1 & i1 > i2 & (i1, r1a) : sa..contents & 0 <= i1 & i1 <
                sa..csize))" */
4275        /*: assume "0 <= i2 & i2 < sa..csize" */
4276        Object r2a = sa.remove_at(i2);
4277
4278        /*: assume "0 <= i2 & i2 < sb..csize" */
4279        Object r2b = sb.remove_at(i2);
4280        /*: assume "0 <= i1 & i1 < sb..csize" */
4281        Object r1b = sb.remove_at(i1);
4282
4283        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
4284    }
4285
4286    static void remove_at_remove_at_post_s_122(ArrayList sa, ArrayList sb, int i1, int
            i2)
4287    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4288                    sa..contents = sb..contents & sa..csize = sb..csize"
4289        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4290        ensures "True" */
4291    {
4292        /*: assume "0 <= i1 & i1 < sa..csize" */
4293        Object r1a = sa.remove_at(i1);
4294        /*: assume "0 <= i2 & i2 < sa..csize" */
4295        Object r2a = sa.remove_at(i2);
4296        /*: assume "(i1 < i2 & (i2 - 1, r2a) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
                sa..csize) | (i1 = i2 & r1a = r2a) | (sa..csize + 1 > i1 & i1 > i2 & (i1 -
                1, r1a) : sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize)" */
4297
4298        Object r2b = sb.remove_at(i2);
4299        Object r1b = sb.remove_at(i1);
4300
4301        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
4302    }
4303
4304    static void remove_at_remove_at_post_c_122(ArrayList sa, ArrayList sb, int i1, int
            i2)
4305    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4306                    sa..contents = sb..contents & sa..csize = sb..csize"
4307        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4308        ensures "True" */
4309    {
4310        /*: assume "0 <= i1 & i1 < sa..csize" */
```

```
        Object r1a = sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);
        /*: assume "~((i1 < i2 & (i2 - 1, r2a) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
            sa..csize) | (i1 = i2 & r1a = r2a) | (sa..csize + 1 > i1 & i1 > i2 & (i1 -
            1, r1a) : sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize))" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.remove_at(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void remove_at_remove_at_pre_s_123(ArrayList sa, ArrayList sb, int i1, int
        i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
            sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
            <= i1 + 1 & i1 + 1 < sa..csize)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        sb.remove_at(i2);
        Object r1b = sb.remove_at(i1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_remove_at_pre_c_123(ArrayList sa, ArrayList sb, int i1, int
        i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
            sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
            <= i1 + 1 & i1 + 1 < sa..csize))" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.remove_at(i1);

```

```
        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void remove_at_remove_at_between_s_124(ArrayList sa, ArrayList sb, int i1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
            sa..csize) | (i1 = i2 & (i2, r1a) : sa..contents & 0 <= i2 & i2 < sa..csize)
            | (sa..csize > i1 & i1 > i2 & (i1, r1a) : sa..contents & 0 <= i1 & i1 <
            sa..csize)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        sb.remove_at(i2);
        Object r1b = sb.remove_at(i1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_remove_at_between_c_124(ArrayList sa, ArrayList sb, int i1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
            sa..csize) | (i1 = i2 & (i2, r1a) : sa..contents & 0 <= i2 & i2 < sa..csize)
            | (sa..csize > i1 & i1 > i2 & (i1, r1a) : sa..contents & 0 <= i1 & i1 <
            sa..csize))" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.remove_at(i1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void remove_at_remove_at_post_s_125(ArrayList sa, ArrayList sb, int i1, int
        i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
        /*: ghost specvar sa__csize :: "int" = "sa..csize" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
```

```
4411        sa.remove_at(i2);
4412        /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
                sa__contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
                sa__csize) | (i1 = i2 & (i2, r1a) : sa__contents & 0 <= i2 & i2 < sa__csize)
                | (sa..csize + 1 > i1 & i1 > i2 & (i1 - 1, r1a) : sa..contents & 0 <= i1 - 1
                & i1 - 1 < sa..csize)" */
4413
4414        sb.remove_at(i2);
4415        Object r1b = sb.remove_at(i1);
4416
4417        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
4418    }
4419
4420    static void remove_at_remove_at_post_c_125(ArrayList sa, ArrayList sb, int i1, int
            i2)
4421    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4422                    sa..contents = sb..contents & sa..csize = sb..csize"
4423        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4424        ensures "True" */
4425    {
4426        /*: assume "0 <= i1 & i1 < sa..csize" */
4427        Object r1a = sa.remove_at(i1);
4428        /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
4429        /*: ghost specvar sa__csize :: "int" = "sa..csize" */
4430        /*: assume "0 <= i2 & i2 < sa..csize" */
4431        sa.remove_at(i2);
4432        /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
                sa__contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
                sa__csize) | (i1 = i2 & (i2, r1a) : sa__contents & 0 <= i2 & i2 < sa__csize)
                | (sa..csize + 1 > i1 & i1 > i2 & (i1 - 1, r1a) : sa..contents & 0 <= i1 - 1
                & i1 - 1 < sa..csize))" */
4433
4434        /*: assume "0 <= i2 & i2 < sb..csize" */
4435        sb.remove_at(i2);
4436        /*: assume "0 <= i1 & i1 < sb..csize" */
4437        Object r1b = sb.remove_at(i1);
4438
4439        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
4440    }
4441
4442    static void remove_at_set_pre_s_126(ArrayList sa, ArrayList sb, int i1, int i2,
            Object v2)
4443    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4444                    sa..contents = sb..contents & sa..csize = sb..csize"
4445        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4446        ensures "True" */
4447    {
4448        /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
                sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) : sa..contents & 0
                <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & (ALL
                v. ((i1, v) : sa..contents) = ((i2 + 1, v) : sa..contents)) & (i1, v2) :
                sa..contents & (i2 + 1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <=
                i2 + 1 & i2 + 1 < sa..csize) | (i1 > i2)" */
4449        /*: assume "0 <= i1 & i1 < sa..csize" */
4450        Object r1a = sa.remove_at(i1);
4451        /*: assume "0 <= i2 & i2 < sa..csize" */
4452        Object r2a = sa.set(i2, v2);
4453
4454        Object r2b = sb.set(i2, v2);
4455        Object r1b = sb.remove_at(i1);
4456
4457        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
4458    }
```

```
4459
4460    static void remove_at_set_pre_c_126(ArrayList sa, ArrayList sb, int i1, int i2,
            Object v2)
4461    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4462                 sa..contents = sb..contents & sa..csize = sb..csize"
4463        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4464        ensures "True" */
4465    {
4466        /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
                sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) : sa..contents & 0
                <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & (ALL
                v. ((i1, v) : sa..contents) = ((i2 + 1, v) : sa..contents)) & (i1, v2) :
                sa..contents & (i2 + 1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <=
                i2 + 1 & i2 + 1 < sa..csize) | (i1 > i2))" */
4467        /*: assume "0 <= i1 & i1 < sa..csize" */
4468        Object r1a = sa.remove_at(i1);
4469        /*: assume "0 <= i2 & i2 < sa..csize" */
4470        Object r2a = sa.set(i2, v2);
4471
4472        /*: assume "0 <= i2 & i2 < sb..csize" */
4473        Object r2b = sb.set(i2, v2);
4474        /*: assume "0 <= i1 & i1 < sb..csize" */
4475        Object r1b = sb.remove_at(i1);
4476
4477        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
4478    }
4479
4480    static void remove_at_set_between_s_127(ArrayList sa, ArrayList sb, int i1, int i2,
            Object v2)
4481    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4482                 sa..contents = sb..contents & sa..csize = sb..csize"
4483        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4484        ensures "True" */
4485    {
4486        /*: assume "0 <= i1 & i1 < sa..csize" */
4487        Object r1a = sa.remove_at(i1);
4488        /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
                sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa..contents & 0
                <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize) | (i1 = i2 & (i2,
                r1a) : sa..contents & r1a = v2 & (i2, v2) : sa..contents & 0 <= i2 & i2 <
                sa..csize) | i1 > i2" */
4489        /*: assume "0 <= i2 & i2 < sa..csize" */
4490        Object r2a = sa.set(i2, v2);
4491
4492        Object r2b = sb.set(i2, v2);
4493        Object r1b = sb.remove_at(i1);
4494
4495        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
4496    }
4497
4498    static void remove_at_set_between_c_127(ArrayList sa, ArrayList sb, int i1, int i2,
            Object v2)
4499    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4500                 sa..contents = sb..contents & sa..csize = sb..csize"
4501        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4502        ensures "True" */
4503    {
4504        /*: assume "0 <= i1 & i1 < sa..csize" */
4505        Object r1a = sa.remove_at(i1);
4506        /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
                sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa..contents & 0
                <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize) | (i1 = i2 & (i2,
```

```
        r1a) : sa..contents & r1a = v2 & (i2, v2) : sa..contents & 0 <= i2 & i2 <
        sa..csize) | i1 > i2)" */
    /*: assume "0 <= i2 & i2 < sa..csize" */
    Object r2a = sa.set(i2, v2);

    /*: assume "0 <= i2 & i2 < sb..csize" */
    Object r2b = sb.set(i2, v2);
    /*: assume "0 <= i1 & i1 < sb..csize" */
    Object r1b = sb.remove_at(i1);

    /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
}

static void remove_at_set_post_s_128(ArrayList sa, ArrayList sb, int i1, int i2,
    Object v2)
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
              sa..contents = sb..contents & sa..csize = sb..csize"
    modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
    ensures "True" */
{
    /*: assume "0 <= i1 & i1 < sa..csize" */
    Object r1a = sa.remove_at(i1);
    /*: assume "0 <= i2 & i2 < sa..csize" */
    Object r2a = sa.set(i2, v2);
    /*: assume "(i1 < i2 & (i2 - 1, r2a) : sa..contents & (i2 - 1, v2) :
        sa..contents & r2a = v2 & 0 <= i2 - 1 & i2 - 1 < sa..csize) | (i1 = i2 & r1a
        = r2a & r1a = v2 & r2a = v2) | i1 > i2" */

    Object r2b = sb.set(i2, v2);
    Object r1b = sb.remove_at(i1);

    /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
}

static void remove_at_set_post_c_128(ArrayList sa, ArrayList sb, int i1, int i2,
    Object v2)
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
              sa..contents = sb..contents & sa..csize = sb..csize"
    modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
    ensures "True" */
{
    /*: assume "0 <= i1 & i1 < sa..csize" */
    Object r1a = sa.remove_at(i1);
    /*: assume "0 <= i2 & i2 < sa..csize" */
    Object r2a = sa.set(i2, v2);
    /*: assume "~((i1 < i2 & (i2 - 1, r2a) : sa..contents & (i2 - 1, v2) :
        sa..contents & r2a = v2 & 0 <= i2 - 1 & i2 - 1 < sa..csize) | (i1 = i2 & r1a
        = r2a & r1a = v2 & r2a = v2) | i1 > i2)" */

    /*: assume "0 <= i2 & i2 < sb..csize" */
    Object r2b = sb.set(i2, v2);
    /*: assume "0 <= i1 & i1 < sb..csize" */
    Object r1b = sb.remove_at(i1);

    /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
}

static void remove_at_set_pre_s_129(ArrayList sa, ArrayList sb, int i1, int i2,
    Object v2)
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
              sa..contents = sb..contents & sa..csize = sb..csize"
    modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
```

```
        ensures "True" */
    {
        /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) : sa..contents & 0
            <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & (ALL
            v. ((i1, v) : sa..contents) = ((i2 + 1, v) : sa..contents)) & (i1, v2) :
            sa..contents & (i2 + 1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <=
            i2 + 1 & i2 + 1 < sa..csize) | (i1 > i2)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        sb.set(i2, v2);
        Object r1b = sb.remove_at(i1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_set_pre_c_129(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) : sa..contents & 0
            <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & (ALL
            v. ((i1, v) : sa..contents) = ((i2 + 1, v) : sa..contents)) & (i1, v2) :
            sa..contents & (i2 + 1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <=
            i2 + 1 & i2 + 1 < sa..csize) | (i1 > i2))" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.remove_at(i1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void remove_at_set_between_s_130(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa..contents & 0
            <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize) | (i1 = i2 & (i2,
            r1a) : sa..contents & r1a = v2 & (i2, v2) : sa..contents & 0 <= i2 & i2 <
            sa..csize) | i1 > i2" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        sb.set(i2, v2);
        Object r1b = sb.remove_at(i1);
```

```
4608
4609        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
4610    }
4611
4612    static void remove_at_set_between_c_130(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
4613    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4614                  sa..contents = sb..contents & sa..csize = sb..csize"
4615        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4616        ensures "True" */
4617    {
4618        /*: assume "0 <= i1 & i1 < sa..csize" */
4619        Object r1a = sa.remove_at(i1);
4620        /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa..contents & 0
            <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize) | (i1 = i2 & (i2,
            r1a) : sa..contents & r1a = v2 & (i2, v2) : sa..contents & 0 <= i2 & i2 <
            sa..csize) | i1 > i2)" */
4621        /*: assume "0 <= i2 & i2 < sa..csize" */
4622        sa.set(i2, v2);
4623
4624        /*: assume "0 <= i2 & i2 < sb..csize" */
4625        sb.set(i2, v2);
4626        /*: assume "0 <= i1 & i1 < sb..csize" */
4627        Object r1b = sb.remove_at(i1);
4628
4629        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
4630    }
4631
4632    static void remove_at_set_post_s_131(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
4633    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4634                  sa..contents = sb..contents & sa..csize = sb..csize"
4635        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4636        ensures "True" */
4637    {
4638        /*: assume "0 <= i1 & i1 < sa..csize" */
4639        Object r1a = sa.remove_at(i1);
4640        /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
4641        /*: ghost specvar sa__csize :: "int" = "sa..csize" */
4642        /*: assume "0 <= i2 & i2 < sa..csize" */
4643        sa.set(i2, v2);
4644        /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa__contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa__contents & 0
            <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa__csize) | (i1 = i2 & (i2,
            r1a) : sa__contents & r1a = v2 & (i2, v2) : sa__contents & 0 <= i2 & i2 <
            sa__csize) | i1 > i2" */
4645
4646        sb.set(i2, v2);
4647        Object r1b = sb.remove_at(i1);
4648
4649        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
4650    }
4651
4652    static void remove_at_set_post_c_131(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
4653    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4654                  sa..contents = sb..contents & sa..csize = sb..csize"
4655        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4656        ensures "True" */
4657    {
4658        /*: assume "0 <= i1 & i1 < sa..csize" */
4659        Object r1a = sa.remove_at(i1);
4660        /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
```

```
        /*: ghost specvar sa__csize :: "int" = "sa..csize" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);
        /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa__contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa__contents & 0
            <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa__csize) | (i1 = i2 & (i2,
            r1a) : sa__contents & r1a = v2 & (i2, v2) : sa__contents & 0 <= i2 & i2 <
            sa__csize) | i1 > i2)" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.remove_at(i1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void remove_at_size_pre_s_132(ArrayList sa, ArrayList sb, int i1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "False" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        int r2a = sa.size();

        int r2b = sb.size();
        Object r1b = sb.remove_at(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void remove_at_size_pre_c_132(ArrayList sa, ArrayList sb, int i1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~(False)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        int r2a = sa.size();

        int r2b = sb.size();
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.remove_at(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void remove_at_size_between_s_133(ArrayList sa, ArrayList sb, int i1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "False" */
        int r2a = sa.size();
```

```
4719
4720        int r2b = sb.size();
4721        Object r1b = sb.remove_at(i1);
4722
4723        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
4724    }
4725
4726    static void remove_at_size_between_c_133(ArrayList sa, ArrayList sb, int i1)
4727    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4728                  sa..contents = sb..contents & sa..csize = sb..csize"
4729        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4730        ensures "True" */
4731    {
4732        /*: assume "0 <= i1 & i1 < sa..csize" */
4733        Object r1a = sa.remove_at(i1);
4734        /*: assume "~(False)" */
4735        int r2a = sa.size();
4736
4737        int r2b = sb.size();
4738        /*: assume "0 <= i1 & i1 < sb..csize" */
4739        Object r1b = sb.remove_at(i1);
4740
4741        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
4742    }
4743
4744    static void remove_at_size_post_s_134(ArrayList sa, ArrayList sb, int i1)
4745    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4746                  sa..contents = sb..contents & sa..csize = sb..csize"
4747        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4748        ensures "True" */
4749    {
4750        /*: assume "0 <= i1 & i1 < sa..csize" */
4751        Object r1a = sa.remove_at(i1);
4752        int r2a = sa.size();
4753        /*: assume "False" */
4754
4755        int r2b = sb.size();
4756        Object r1b = sb.remove_at(i1);
4757
4758        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
4759    }
4760
4761    static void remove_at_size_post_c_134(ArrayList sa, ArrayList sb, int i1)
4762    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4763                  sa..contents = sb..contents & sa..csize = sb..csize"
4764        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4765        ensures "True" */
4766    {
4767        /*: assume "0 <= i1 & i1 < sa..csize" */
4768        Object r1a = sa.remove_at(i1);
4769        int r2a = sa.size();
4770        /*: assume "~(False)" */
4771
4772        int r2b = sb.size();
4773        /*: assume "0 <= i1 & i1 < sb..csize" */
4774        Object r1b = sb.remove_at(i1);
4775
4776        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
4777    }
4778
```

```
4779    static void remove_at_add_at_pre_s_135(ArrayList sa, ArrayList sb, int i1, int i2,
            Object v2)
4780    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4781                sa..contents = sb..contents & sa..csize = sb..csize"
4782        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
4783        ensures "True" */
4784    {
4785        /*: assume "(i1 < i2 & (i2, v2) : sa..contents & 0 <= i2 & i2 < sa..csize) |
                (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize) | (i1 > i2 &
                (ALL v. ((i1 - 1, v) : sa..contents) = ((i1, v) : sa..contents)) & 0 <= i1 -
                1 & i1 - 1 < sa..csize & 0 <= i1 & i1 < sa..csize)" */
4786        /*: assume "0 <= i1 & i1 < sa..csize" */
4787        sa.remove_at(i1);
4788        /*: assume "0 <= i2 & i2 <= sa..csize" */
4789        sa.add_at(i2, v2);
4790
4791        sb.add_at(i2, v2);
4792        sb.remove_at(i1);
4793
4794        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
4795    }
4796
4797    static void remove_at_add_at_pre_c_135(ArrayList sa, ArrayList sb, int i1, int i2,
            Object v2)
4798    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4799                sa..contents = sb..contents & sa..csize = sb..csize"
4800        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
4801        ensures "True" */
4802    {
4803        /*: assume "~((i1 < i2 & (i2, v2) : sa..contents & 0 <= i2 & i2 < sa..csize) |
                (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize) | (i1 > i2 &
                (ALL v. ((i1 - 1, v) : sa..contents) = ((i1, v) : sa..contents)) & 0 <= i1 -
                1 & i1 - 1 < sa..csize & 0 <= i1 & i1 < sa..csize))" */
4804        /*: assume "0 <= i1 & i1 < sa..csize" */
4805        sa.remove_at(i1);
4806        /*: assume "0 <= i2 & i2 <= sa..csize" */
4807        sa.add_at(i2, v2);
4808
4809        /*: assume "0 <= i2 & i2 <= sb..csize" */
4810        sb.add_at(i2, v2);
4811        /*: assume "0 <= i1 & i1 < sb..csize" */
4812        sb.remove_at(i1);
4813
4814        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
4815    }
4816
4817    static void remove_at_add_at_between_s_136(ArrayList sa, ArrayList sb, int i1, int
            i2, Object v2)
4818    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4819                sa..contents = sb..contents & sa..csize = sb..csize"
4820        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
4821        ensures "True" */
4822    {
4823        /*: assume "0 <= i1 & i1 < sa..csize" */
4824        sa.remove_at(i1);
4825        /*: assume "(i1 < i2 & (i2 - 1, v2) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
                sa..csize) | (i1 = i2 & (i1, v2) : sa..(old contents) & 0 <= i1 & i1 <
                sa..(old csize)) | (i1 > i2 & (ALL v. ((i1 - 1, v) : sa..contents) = ((i1,
                v) : sa..(old contents))) & 0 <= i1 - 1 & i1 - 1 < sa..csize & 0 <= i1 & i1
                < sa..(old csize))" */
4826        /*: assume "0 <= i2 & i2 <= sa..csize" */
4827        sa.add_at(i2, v2);
```

```
4828
4829        sb.add_at(i2, v2);
4830        sb.remove_at(i1);
4831
4832        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
4833    }
4834
4835    static void remove_at_add_at_between_c_136(ArrayList sa, ArrayList sb, int i1, int
        i2, Object v2)
4836    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4837                sa..contents = sb..contents & sa..csize = sb..csize"
4838        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
4839        ensures "True" */
4840    {
4841        /*: assume "0 <= i1 & i1 < sa..csize" */
4842        sa.remove_at(i1);
4843        /*: assume "~((i1 < i2 & (i2 - 1, v2) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
            sa..csize) | (i1 = i2 & (i1, v2) : sa..(old contents) & 0 <= i1 & i1 <
            sa..(old csize)) | (i1 > i2 & (ALL v. ((i1 - 1, v) : sa..contents) = ((i1,
            v) : sa..(old contents))) & 0 <= i1 - 1 & i1 - 1 < sa..csize & 0 <= i1 & i1
            < sa..(old csize)))" */
4844        /*: assume "0 <= i2 & i2 <= sa..csize" */
4845        sa.add_at(i2, v2);
4846
4847        /*: assume "0 <= i2 & i2 <= sb..csize" */
4848        sb.add_at(i2, v2);
4849        /*: assume "0 <= i1 & i1 < sb..csize" */
4850        sb.remove_at(i1);
4851
4852        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
4853    }
4854
4855    static void remove_at_add_at_post_s_137(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
4856    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4857                sa..contents = sb..contents & sa..csize = sb..csize"
4858        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
4859        ensures "True" */
4860    {
4861        /*: assume "0 <= i1 & i1 < sa..csize" */
4862        sa.remove_at(i1);
4863        /*: assume "0 <= i2 & i2 <= sa..csize" */
4864        sa.add_at(i2, v2);
4865        /*: assume "(i1 < i2 & (i2 - 1, v2) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
            sa..csize) | (i1 = i2 & (i1, v2) : sa..(old contents) & 0 <= i1 & i1 <
            sa..(old csize)) | (i1 > i2 & (ALL v. ((i1, v) : sa..contents) = ((i1, v) :
            sa..(old contents))) & 0 <= i1 & i1 < sa..csize & 0 <= i1 & i1 < sa..(old
            csize))" */
4866
4867        sb.add_at(i2, v2);
4868        sb.remove_at(i1);
4869
4870        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
4871    }
4872
4873    static void remove_at_add_at_post_c_137(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
4874    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4875                sa..contents = sb..contents & sa..csize = sb..csize"
4876        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
4877        ensures "True" */
4878    {
```

```
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);
        /*: assume "~((i1 < i2 & (i2 - 1, v2) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
            sa..csize) | (i1 = i2 & (i1, v2) : sa..(old contents) & 0 <= i1 & i1 <
            sa..(old csize)) | (i1 > i2 & (ALL v. ((i1, v) : sa..contents) = ((i1, v) :
            sa..(old contents))) & 0 <= i1 & i1 < sa..csize & 0 <= i1 & i1 < sa..(old
            csize)))" */

        /*: assume "0 <= i2 & i2 <= sb..csize" */
        sb.add_at(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.remove_at(i1);

        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
    }

    static void remove_at_get_pre_s_138(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | i1 > i2" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        Object r2b = sb.get(i2);
        sb.remove_at(i1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_get_pre_c_138(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | i1 > i2)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.get(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.remove_at(i1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }
```

```
4931    static void remove_at_get_between_s_139(ArrayList sa, ArrayList sb, int i1, int i2)
4932    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4933                sa..contents = sb..contents & sa..csize = sb..csize"
4934        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4935        ensures "True" */
4936    {
4937        /*: assume "0 <= i1 & i1 < sa..csize" */
4938        sa.remove_at(i1);
4939        /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
                sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
                sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..(old contents)) = ((i2, v) :
                sa..contents)) & 0 <= i1 & i1 < sa..(old csize) & 0 <= i2 & i2 < sa..csize)
                | i1 > i2" */
4940        /*: assume "0 <= i2 & i2 < sa..csize" */
4941        Object r2a = sa.get(i2);
4942
4943        Object r2b = sb.get(i2);
4944        sb.remove_at(i1);
4945
4946        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
4947    }
4948
4949    static void remove_at_get_between_c_139(ArrayList sa, ArrayList sb, int i1, int i2)
4950    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4951                sa..contents = sb..contents & sa..csize = sb..csize"
4952        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4953        ensures "True" */
4954    {
4955        /*: assume "0 <= i1 & i1 < sa..csize" */
4956        sa.remove_at(i1);
4957        /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
                sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
                sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..(old contents)) = ((i2, v) :
                sa..contents)) & 0 <= i1 & i1 < sa..(old csize) & 0 <= i2 & i2 < sa..csize)
                | i1 > i2)" */
4958        /*: assume "0 <= i2 & i2 < sa..csize" */
4959        Object r2a = sa.get(i2);
4960
4961        /*: assume "0 <= i2 & i2 < sb..csize" */
4962        Object r2b = sb.get(i2);
4963        /*: assume "0 <= i1 & i1 < sb..csize" */
4964        sb.remove_at(i1);
4965
4966        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
4967    }
4968
4969    static void remove_at_get_post_s_140(ArrayList sa, ArrayList sb, int i1, int i2)
4970    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4971                sa..contents = sb..contents & sa..csize = sb..csize"
4972        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4973        ensures "True" */
4974    {
4975        /*: assume "0 <= i1 & i1 < sa..csize" */
4976        sa.remove_at(i1);
4977        /*: assume "0 <= i2 & i2 < sa..csize" */
4978        Object r2a = sa.get(i2);
4979        /*: assume "(i1 < i2 & (i2 - 1, r2a) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
                sa..csize) | (i1 = i2 & (i1, r2a) : sa..(old contents) & 0 <= i1 & i1 <
                sa..(old csize)) | i1 > i2" */
4980
4981        Object r2b = sb.get(i2);
4982        sb.remove_at(i1);
4983
4984        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
```

```
4985    }
4986
4987    static void remove_at_get_post_c_140(ArrayList sa, ArrayList sb, int i1, int i2)
4988    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4989                sa..contents = sb..contents & sa..csize = sb..csize"
4990        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4991        ensures "True" */
4992    {
4993        /*: assume "0 <= i1 & i1 < sa..csize" */
4994        sa.remove_at(i1);
4995        /*: assume "0 <= i2 & i2 < sa..csize" */
4996        Object r2a = sa.get(i2);
4997        /*: assume "~((i1 < i2 & (i2 - 1, r2a) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
                sa..csize) | (i1 = i2 & (i1, r2a) : sa..(old contents) & 0 <= i1 & i1 <
                sa..(old csize)) | i1 > i2)" */
4998
4999        /*: assume "0 <= i2 & i2 < sb..csize" */
5000        Object r2b = sb.get(i2);
5001        /*: assume "0 <= i1 & i1 < sb..csize" */
5002        sb.remove_at(i1);
5003
5004        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
5005    }
5006
5007    static void remove_at_indexOf_pre_s_141(ArrayList sa, ArrayList sb, int i1, Object
            v2)
5008    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5009                sa..contents = sb..contents & sa..csize = sb..csize"
5010        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5011        ensures "True" */
5012    {
5013        /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
                (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) : sa..contents
                & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & i1
                < sa..csize - 1 & (i1 + 1, v2) : sa..contents & 0 <= i1 + 1 & i1 + 1 <
                sa..csize)" */
5014        /*: assume "0 <= i1 & i1 < sa..csize" */
5015        sa.remove_at(i1);
5016        int r2a = sa.indexOf(v2);
5017
5018        int r2b = sb.indexOf(v2);
5019        sb.remove_at(i1);
5020
5021        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
5022    }
5023
5024    static void remove_at_indexOf_pre_c_141(ArrayList sa, ArrayList sb, int i1, Object
            v2)
5025    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5026                sa..contents = sb..contents & sa..csize = sb..csize"
5027        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5028        ensures "True" */
5029    {
5030        /*: assume "~(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
                (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) : sa..contents
                & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & i1
                < sa..csize - 1 & (i1 + 1, v2) : sa..contents & 0 <= i1 + 1 & i1 + 1 <
                sa..csize))" */
5031        /*: assume "0 <= i1 & i1 < sa..csize" */
5032        sa.remove_at(i1);
5033        int r2a = sa.indexOf(v2);
5034
5035        int r2b = sb.indexOf(v2);
5036        /*: assume "0 <= i1 & i1 < sb..csize" */
```

```
5037        sb.remove_at(i1);
5038
5039        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
5040    }
5041
5042    static void remove_at_indexOf_between_s_142(ArrayList sa, ArrayList sb, int i1,
            Object v2)
5043    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5044                    sa..contents = sb..contents & sa..csize = sb..csize"
5045        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5046        ensures "True" */
5047    {
5048        /*: assume "0 <= i1 & i1 < sa..csize" */
5049        sa.remove_at(i1);
5050        /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & (i1,
                v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | (EX i. (i, v2)
                : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) : sa..contents & 0 <= i
                & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & (i1, v2) :
                sa..(old contents) & 0 <= i1 & i1 < sa..(old csize) & i1 < sa..csize)" */
5051        int r2a = sa.indexOf(v2);
5052
5053        int r2b = sb.indexOf(v2);
5054        sb.remove_at(i1);
5055
5056        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
5057    }
5058
5059    static void remove_at_indexOf_between_c_142(ArrayList sa, ArrayList sb, int i1,
            Object v2)
5060    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5061                    sa..contents = sb..contents & sa..csize = sb..csize"
5062        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5063        ensures "True" */
5064    {
5065        /*: assume "0 <= i1 & i1 < sa..csize" */
5066        sa.remove_at(i1);
5067        /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & (i1,
                v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | (EX i. (i, v2)
                : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) : sa..contents & 0 <= i
                & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & (i1, v2) :
                sa..(old contents) & 0 <= i1 & i1 < sa..(old csize) & i1 < sa..csize))" */
5068        int r2a = sa.indexOf(v2);
5069
5070        int r2b = sb.indexOf(v2);
5071        /*: assume "0 <= i1 & i1 < sb..csize" */
5072        sb.remove_at(i1);
5073
5074        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
5075    }
5076
5077    static void remove_at_indexOf_post_s_143(ArrayList sa, ArrayList sb, int i1, Object
            v2)
5078    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5079                    sa..contents = sb..contents & sa..csize = sb..csize"
5080        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5081        ensures "True" */
5082    {
5083        /*: assume "0 <= i1 & i1 < sa..csize" */
5084        sa.remove_at(i1);
5085        int r2a = sa.indexOf(v2);
5086        /*: assume "(r2a < 0 & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old
                csize)) | (0 <= r2a & r2a < i1) | (r2a = i1 & (i1, v2) : sa..(old contents)
                & 0 <= i1 & i1 < sa..(old csize) & i1 < sa..csize)" */
```

```
5087
5088        int r2b = sb.indexOf(v2);
5089        sb.remove_at(i1);
5090
5091        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
5092    }
5093
5094    static void remove_at_indexOf_post_c_143(ArrayList sa, ArrayList sb, int i1, Object
        v2)
5095    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5096                sa..contents = sb..contents & sa..csize = sb..csize"
5097        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5098        ensures "True" */
5099    {
5100        /*: assume "0 <= i1 & i1 < sa..csize" */
5101        sa.remove_at(i1);
5102        int r2a = sa.indexOf(v2);
5103        /*: assume "~((r2a < 0 & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 <
                sa..(old csize)) | (0 <= r2a & r2a < i1) | (r2a = i1 & (i1, v2) : sa..(old
                contents) & 0 <= i1 & i1 < sa..(old csize) & i1 < sa..csize))" */
5104
5105        int r2b = sb.indexOf(v2);
5106        /*: assume "0 <= i1 & i1 < sb..csize" */
5107        sb.remove_at(i1);
5108
5109        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
5110    }
5111
5112    static void remove_at_lastIndexOf_pre_s_144(ArrayList sa, ArrayList sb, int i1,
        Object v2)
5113    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5114                sa..contents = sb..contents & sa..csize = sb..csize"
5115        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5116        ensures "True" */
5117    {
5118        /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | ((EX i.
                (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) : sa..contents &
                i1 <= i & i < sa..csize))" */
5119        /*: assume "0 <= i1 & i1 < sa..csize" */
5120        sa.remove_at(i1);
5121        int r2a = sa.lastIndexOf(v2);
5122
5123        int r2b = sb.lastIndexOf(v2);
5124        sb.remove_at(i1);
5125
5126        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
5127    }
5128
5129    static void remove_at_lastIndexOf_pre_c_144(ArrayList sa, ArrayList sb, int i1,
        Object v2)
5130    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5131                sa..contents = sb..contents & sa..csize = sb..csize"
5132        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5133        ensures "True" */
5134    {
5135        /*: assume "~(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | ((EX
                i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
                sa..contents & i1 <= i & i < sa..csize)))" */
5136        /*: assume "0 <= i1 & i1 < sa..csize" */
5137        sa.remove_at(i1);
5138        int r2a = sa.lastIndexOf(v2);
5139
5140        int r2b = sb.lastIndexOf(v2);
5141        /*: assume "0 <= i1 & i1 < sb..csize" */
```

```
5142        sb.remove_at(i1);
5143
5144        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
5145    }
5146
5147    static void remove_at_lastIndexOf_between_s_145(ArrayList sa, ArrayList sb, int i1,
            Object v2)
5148    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5149                  sa..contents = sb..contents & sa..csize = sb..csize"
5150        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5151        ensures "True" */
5152    {
5153        /*: assume "0 <= i1 & i1 < sa..csize" */
5154        sa.remove_at(i1);
5155        /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & (i1,
                v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | ((EX i. (i,
                v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) : sa..contents & i1
                <= i & i < sa..csize) & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 <
                sa..(old csize))" */
5156        int r2a = sa.lastIndexOf(v2);
5157
5158        int r2b = sb.lastIndexOf(v2);
5159        sb.remove_at(i1);
5160
5161        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
5162    }
5163
5164    static void remove_at_lastIndexOf_between_c_145(ArrayList sa, ArrayList sb, int i1,
            Object v2)
5165    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5166                  sa..contents = sb..contents & sa..csize = sb..csize"
5167        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5168        ensures "True" */
5169    {
5170        /*: assume "0 <= i1 & i1 < sa..csize" */
5171        sa.remove_at(i1);
5172        /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & (i1,
                v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | ((EX i. (i,
                v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) : sa..contents & i1
                <= i & i < sa..csize) & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 <
                sa..(old csize)))" */
5173        int r2a = sa.lastIndexOf(v2);
5174
5175        int r2b = sb.lastIndexOf(v2);
5176        /*: assume "0 <= i1 & i1 < sb..csize" */
5177        sb.remove_at(i1);
5178
5179        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
5180    }
5181
5182    static void remove_at_lastIndexOf_post_s_146(ArrayList sa, ArrayList sb, int i1,
            Object v2)
5183    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5184                  sa..contents = sb..contents & sa..csize = sb..csize"
5185        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5186        ensures "True" */
5187    {
5188        /*: assume "0 <= i1 & i1 < sa..csize" */
5189        sa.remove_at(i1);
5190        int r2a = sa.lastIndexOf(v2);
5191        /*: assume "(r2a < 0 & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old
                csize)) | (0 <= r2a & r2a < i1 & (i1, v2) ~: sa..(old contents) & 0 <= i1 &
                i1 < sa..(old csize))" */
```

```
5192
5193        int r2b = sb.lastIndexOf(v2);
5194        sb.remove_at(i1);
5195
5196        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
5197    }
5198
5199    static void remove_at_lastIndexOf_post_c_146(ArrayList sa, ArrayList sb, int i1,
        Object v2)
5200    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5201                sa..contents = sb..contents & sa..csize = sb..csize"
5202        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5203        ensures "True" */
5204    {
5205        /*: assume "0 <= i1 & i1 < sa..csize" */
5206        sa.remove_at(i1);
5207        int r2a = sa.lastIndexOf(v2);
5208        /*: assume "~((r2a < 0 & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 <
            sa..(old csize)) | (0 <= r2a & r2a < i1 & (i1, v2) ~: sa..(old contents) & 0
            <= i1 & i1 < sa..(old csize)))" */
5209
5210        int r2b = sb.lastIndexOf(v2);
5211        /*: assume "0 <= i1 & i1 < sb..csize" */
5212        sb.remove_at(i1);
5213
5214        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
5215    }
5216
5217    static void remove_at_remove_at_pre_s_147(ArrayList sa, ArrayList sb, int i1, int
        i2)
5218    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5219                sa..contents = sb..contents & sa..csize = sb..csize"
5220        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5221        ensures "True" */
5222    {
5223        /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
            sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
            <= i1 + 1 & i1 + 1 < sa..csize)" */
5224        /*: assume "0 <= i1 & i1 < sa..csize" */
5225        sa.remove_at(i1);
5226        /*: assume "0 <= i2 & i2 < sa..csize" */
5227        Object r2a = sa.remove_at(i2);
5228
5229        Object r2b = sb.remove_at(i2);
5230        sb.remove_at(i1);
5231
5232        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
5233    }
5234
5235    static void remove_at_remove_at_pre_c_147(ArrayList sa, ArrayList sb, int i1, int
        i2)
5236    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5237                sa..contents = sb..contents & sa..csize = sb..csize"
5238        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5239        ensures "True" */
5240    {
5241        /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
```

```
            sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
            sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
            <= i1 + 1 & i1 + 1 < sa..csize))" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.remove_at(i1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void remove_at_remove_at_between_s_148(ArrayList sa, ArrayList sb, int i1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
            sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..(old contents)) = ((i2, v) :
            sa..contents)) & 0 <= i1 & i1 < sa..(old csize) & 0 <= i2 & i2 < sa..csize)
            | (sa..csize > i1 & i1 > i2 & (ALL v. ((i1, v) : sa..(old contents)) = ((i1,
            v) : sa..contents)) & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 & i1 <
            sa..csize)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);

        Object r2b = sb.remove_at(i2);
        sb.remove_at(i1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_remove_at_between_c_148(ArrayList sa, ArrayList sb, int i1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
            sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..(old contents)) = ((i2, v) :
            sa..contents)) & 0 <= i1 & i1 < sa..(old csize) & 0 <= i2 & i2 < sa..csize)
            | (sa..csize > i1 & i1 > i2 & (ALL v. ((i1, v) : sa..(old contents)) = ((i1,
            v) : sa..contents)) & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 & i1 <
            sa..csize))" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.remove_at(i1);
```

```
5289
5290        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
5291    }
5292
5293    static void remove_at_remove_at_post_s_149(ArrayList sa, ArrayList sb, int i1, int
            i2)
5294    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5295                  sa..contents = sb..contents & sa..csize = sb..csize"
5296        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5297        ensures "True" */
5298    {
5299        /*: assume "0 <= i1 & i1 < sa..csize" */
5300        sa.remove_at(i1);
5301        /*: assume "0 <= i2 & i2 < sa..csize" */
5302        Object r2a = sa.remove_at(i2);
5303        /*: assume "(i1 < i2 & (i2 - 1, r2a) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
                sa..csize) | (i1 = i2 & (i1, r2a) : sa..(old contents) & 0 <= i1 & i1 <
                sa..(old csize)) | (sa..csize + 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
                sa..(old contents)) = ((i1 - 1, v) : sa..contents)) & 0 <= i1 & i1 <
                sa..(old csize) & 0 <= i1 - 1 & i1 - 1 < sa..csize)" */
5304
5305        Object r2b = sb.remove_at(i2);
5306        sb.remove_at(i1);
5307
5308        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
5309    }
5310
5311    static void remove_at_remove_at_post_c_149(ArrayList sa, ArrayList sb, int i1, int
            i2)
5312    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5313                  sa..contents = sb..contents & sa..csize = sb..csize"
5314        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5315        ensures "True" */
5316    {
5317        /*: assume "0 <= i1 & i1 < sa..csize" */
5318        sa.remove_at(i1);
5319        /*: assume "0 <= i2 & i2 < sa..csize" */
5320        Object r2a = sa.remove_at(i2);
5321        /*: assume "~((i1 < i2 & (i2 - 1, r2a) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
                sa..csize) | (i1 = i2 & (i1, r2a) : sa..(old contents) & 0 <= i1 & i1 <
                sa..(old csize)) | (sa..csize + 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
                sa..(old contents)) = ((i1 - 1, v) : sa..contents)) & 0 <= i1 & i1 <
                sa..(old csize) & 0 <= i1 - 1 & i1 - 1 < sa..csize))" */
5322
5323        /*: assume "0 <= i2 & i2 < sb..csize" */
5324        Object r2b = sb.remove_at(i2);
5325        /*: assume "0 <= i1 & i1 < sb..csize" */
5326        sb.remove_at(i1);
5327
5328        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
5329    }
5330
5331    static void remove_at_remove_at_pre_s_150(ArrayList sa, ArrayList sb, int i1, int
            i2)
5332    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5333                  sa..contents = sb..contents & sa..csize = sb..csize"
5334        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5335        ensures "True" */
5336    {
5337        /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
                sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
                sa..csize) | i1 = i2 | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
```

```
            sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
            <= i1 + 1 & i1 + 1 < sa..csize)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        sb.remove_at(i2);
        sb.remove_at(i1);

        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_remove_at_pre_c_150(ArrayList sa, ArrayList sb, int i1, int
        i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | i1 = i2 | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
            sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
            <= i1 + 1 & i1 + 1 < sa..csize))" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.remove_at(i1);

        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
    }

    static void remove_at_remove_at_between_s_151(ArrayList sa, ArrayList sb, int i1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
            sa..csize) | i1 = i2 | (sa..csize > i1 & i1 > i2 & (ALL v. ((i1, v) :
            sa..(old contents)) = ((i1, v) : sa..contents)) & 0 <= i1 & i1 < sa..(old
            csize) & 0 <= i1 & i1 < sa..csize)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        sb.remove_at(i2);
        sb.remove_at(i1);

        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_remove_at_between_c_151(ArrayList sa, ArrayList sb, int i1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
```

```
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
            sa..csize) | i1 = i2 | (sa..csize > i1 & i1 > i2 & (ALL v. ((i1, v) :
            sa..(old contents)) = ((i1, v) : sa..contents)) & 0 <= i1 & i1 < sa..(old
            csize) & 0 <= i1 & i1 < sa..csize))" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.remove_at(i1);

        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
    }

    static void remove_at_remove_at_post_s_152(ArrayList sa, ArrayList sb, int i1, int
        i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
        /*: ghost specvar sa__csize :: "int" = "sa..csize" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);
        /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa__contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
            sa__csize) | i1 = i2 | (sa..csize + 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
            sa..(old contents)) = ((i1 - 1, v) : sa..contents)) & 0 <= i1 & i1 <
            sa..(old csize) & 0 <= i1 - 1 & i1 - 1 < sa..csize)" */

        sb.remove_at(i2);
        sb.remove_at(i1);

        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_remove_at_post_c_152(ArrayList sa, ArrayList sb, int i1, int
        i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
        /*: ghost specvar sa__csize :: "int" = "sa..csize" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);
        /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa__contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
            sa__csize) | i1 = i2 | (sa..csize + 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
            sa..(old contents)) = ((i1 - 1, v) : sa..contents)) & 0 <= i1 & i1 <
            sa..(old csize) & 0 <= i1 - 1 & i1 - 1 < sa..csize))" */

```

```
        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.remove_at(i1);

        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
    }

    static void remove_at_set_pre_s_153(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) : sa..contents & 0
            <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & (ALL
            v. ((i1, v) : sa..contents) = ((i2 + 1, v) : sa..contents)) & (i1, v2) :
            sa..contents & (i2 + 1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <=
            i2 + 1 & i2 + 1 < sa..csize) | (i1 > i2))" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        Object r2b = sb.set(i2, v2);
        sb.remove_at(i1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_set_pre_c_153(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) : sa..contents & 0
            <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & (ALL
            v. ((i1, v) : sa..contents) = ((i2 + 1, v) : sa..contents)) & (i1, v2) :
            sa..contents & (i2 + 1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <=
            i2 + 1 & i2 + 1 < sa..csize) | (i1 > i2))" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.remove_at(i1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void remove_at_set_between_s_154(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
```

```
5492    {
5493        /*: assume "0 <= i1 & i1 < sa..csize" */
5494        sa.remove_at(i1);
5495        /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
                sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa..contents & 0
                <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize) | (i1 = i2 & (ALL
                v. ((i1, v) : sa..(old contents)) = ((i2, v) : sa..contents)) & (i1, v2) :
                sa..(old contents) & (i2, v2) : sa..contents & 0 <= i1 & i1 < sa..(old
                csize) & 0 <= i2 & i2 < sa..csize) | i1 > i2" */
5496        /*: assume "0 <= i2 & i2 < sa..csize" */
5497        Object r2a = sa.set(i2, v2);
5498
5499        Object r2b = sb.set(i2, v2);
5500        sb.remove_at(i1);
5501
5502        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
5503    }
5504
5505    static void remove_at_set_between_c_154(ArrayList sa, ArrayList sb, int i1, int i2,
            Object v2)
5506    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5507                  sa..contents = sb..contents & sa..csize = sb..csize"
5508        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5509        ensures "True" */
5510    {
5511        /*: assume "0 <= i1 & i1 < sa..csize" */
5512        sa.remove_at(i1);
5513        /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
                sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa..contents & 0
                <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize) | (i1 = i2 & (ALL
                v. ((i1, v) : sa..(old contents)) = ((i2, v) : sa..contents)) & (i1, v2) :
                sa..(old contents) & (i2, v2) : sa..contents & 0 <= i1 & i1 < sa..(old
                csize) & 0 <= i2 & i2 < sa..csize) | i1 > i2)" */
5514        /*: assume "0 <= i2 & i2 < sa..csize" */
5515        Object r2a = sa.set(i2, v2);
5516
5517        /*: assume "0 <= i2 & i2 < sb..csize" */
5518        Object r2b = sb.set(i2, v2);
5519        /*: assume "0 <= i1 & i1 < sb..csize" */
5520        sb.remove_at(i1);
5521
5522        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
5523    }
5524
5525    static void remove_at_set_post_s_155(ArrayList sa, ArrayList sb, int i1, int i2,
            Object v2)
5526    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5527                  sa..contents = sb..contents & sa..csize = sb..csize"
5528        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5529        ensures "True" */
5530    {
5531        /*: assume "0 <= i1 & i1 < sa..csize" */
5532        sa.remove_at(i1);
5533        /*: assume "0 <= i2 & i2 < sa..csize" */
5534        Object r2a = sa.set(i2, v2);
5535        /*: assume "(i1 < i2 & (i2 - 1, r2a) : sa..contents & (i2 - 1, v2) :
                sa..contents & r2a = v2 & 0 <= i2 - 1 & i2 - 1 < sa..csize) | (i1 = i2 &
                (i1, r2a) : sa..(old contents) & (i1, v2) : sa..(old contents) & r2a = v2 &
                0 <= i1 & i1 < sa..(old csize)) | i1 > i2" */
5536
5537        Object r2b = sb.set(i2, v2);
5538        sb.remove_at(i1);
5539
5540        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
```

```
5541    }
5542
5543    static void remove_at_set_post_c_155(ArrayList sa, ArrayList sb, int i1, int i2,
            Object v2)
5544    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5545                sa..contents = sb..contents & sa..csize = sb..csize"
5546        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5547        ensures "True" */
5548    {
5549        /*: assume "0 <= i1 & i1 < sa..csize" */
5550        sa.remove_at(i1);
5551        /*: assume "0 <= i2 & i2 < sa..csize" */
5552        Object r2a = sa.set(i2, v2);
5553        /*: assume "~((i1 < i2 & (i2 - 1, r2a) : sa..contents & (i2 - 1, v2) :
                sa..contents & r2a = v2 & 0 <= i2 - 1 & i2 - 1 < sa..csize) | (i1 = i2 &
                (i1, r2a) : sa..(old contents) & (i1, v2) : sa..(old contents) & r2a = v2 &
                0 <= i1 & i1 < sa..(old csize)) | i1 > i2)" */
5554
5555        /*: assume "0 <= i2 & i2 < sb..csize" */
5556        Object r2b = sb.set(i2, v2);
5557        /*: assume "0 <= i1 & i1 < sb..csize" */
5558        sb.remove_at(i1);
5559
5560        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
5561    }
5562
5563    static void remove_at_set_pre_s_156(ArrayList sa, ArrayList sb, int i1, int i2,
            Object v2)
5564    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5565                sa..contents = sb..contents & sa..csize = sb..csize"
5566        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5567        ensures "True" */
5568    {
5569        /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
                sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) : sa..contents & 0
                <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & (i2
                + 1, v2) : sa..contents & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 > i2)" */
5570        /*: assume "0 <= i1 & i1 < sa..csize" */
5571        sa.remove_at(i1);
5572        /*: assume "0 <= i2 & i2 < sa..csize" */
5573        sa.set(i2, v2);
5574
5575        sb.set(i2, v2);
5576        sb.remove_at(i1);
5577
5578        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
5579    }
5580
5581    static void remove_at_set_pre_c_156(ArrayList sa, ArrayList sb, int i1, int i2,
            Object v2)
5582    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5583                sa..contents = sb..contents & sa..csize = sb..csize"
5584        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5585        ensures "True" */
5586    {
5587        /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
                sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) : sa..contents & 0
                <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & (i2
                + 1, v2) : sa..contents & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 > i2))" */
5588        /*: assume "0 <= i1 & i1 < sa..csize" */
5589        sa.remove_at(i1);
5590        /*: assume "0 <= i2 & i2 < sa..csize" */
5591        sa.set(i2, v2);
5592
```

```
        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.remove_at(i1);

        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
    }

    static void remove_at_set_between_s_157(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa..contents & 0
            <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize) | (i1 = i2 & (i2,
            v2) : sa..contents & 0 <= i2 & i2 < sa..csize) | i1 > i2" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        sb.set(i2, v2);
        sb.remove_at(i1);

        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_set_between_c_157(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa..contents & 0
            <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize) | (i1 = i2 & (i2,
            v2) : sa..contents & 0 <= i2 & i2 < sa..csize) | i1 > i2)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.remove_at(i1);

        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
    }

    static void remove_at_set_post_s_158(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
        /*: ghost specvar sa__csize :: "int" = "sa..csize" */
```

```
5649        /*: assume "0 <= i2 & i2 < sa..csize" */
5650        sa.set(i2, v2);
5651        /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
                sa__contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa__contents & 0
                <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa__csize) | (i1 = i2 & (i2,
                v2) : sa__contents & 0 <= i2 & i2 < sa__csize) | i1 > i2" */
5652
5653        sb.set(i2, v2);
5654        sb.remove_at(i1);
5655
5656        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
5657    }
5658
5659    static void remove_at_set_post_c_158(ArrayList sa, ArrayList sb, int i1, int i2,
            Object v2)
5660    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5661                  sa..contents = sb..contents & sa..csize = sb..csize"
5662        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5663        ensures "True" */
5664    {
5665        /*: assume "0 <= i1 & i1 < sa..csize" */
5666        sa.remove_at(i1);
5667        /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
5668        /*: ghost specvar sa__csize :: "int" = "sa..csize" */
5669        /*: assume "0 <= i2 & i2 < sa..csize" */
5670        sa.set(i2, v2);
5671        /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
                sa__contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa__contents & 0
                <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa__csize) | (i1 = i2 & (i2,
                v2) : sa__contents & 0 <= i2 & i2 < sa__csize) | i1 > i2)" */
5672
5673        /*: assume "0 <= i2 & i2 < sb..csize" */
5674        sb.set(i2, v2);
5675        /*: assume "0 <= i1 & i1 < sb..csize" */
5676        sb.remove_at(i1);
5677
5678        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
5679    }
5680
5681    static void remove_at_size_pre_s_159(ArrayList sa, ArrayList sb, int i1)
5682    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5683                  sa..contents = sb..contents & sa..csize = sb..csize"
5684        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5685        ensures "True" */
5686    {
5687        /*: assume "False" */
5688        /*: assume "0 <= i1 & i1 < sa..csize" */
5689        sa.remove_at(i1);
5690        int r2a = sa.size();
5691
5692        int r2b = sb.size();
5693        sb.remove_at(i1);
5694
5695        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
5696    }
5697
5698    static void remove_at_size_pre_c_159(ArrayList sa, ArrayList sb, int i1)
5699    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5700                  sa..contents = sb..contents & sa..csize = sb..csize"
5701        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5702        ensures "True" */
5703    {
5704        /*: assume "~(False)" */
5705        /*: assume "0 <= i1 & i1 < sa..csize" */
5706        sa.remove_at(i1);
```

```
        int r2a = sa.size();

        int r2b = sb.size();
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.remove_at(i1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void remove_at_size_between_s_160(ArrayList sa, ArrayList sb, int i1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "False" */
        int r2a = sa.size();

        int r2b = sb.size();
        sb.remove_at(i1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_size_between_c_160(ArrayList sa, ArrayList sb, int i1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "~(False)" */
        int r2a = sa.size();

        int r2b = sb.size();
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.remove_at(i1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void remove_at_size_post_s_161(ArrayList sa, ArrayList sb, int i1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        int r2a = sa.size();
        /*: assume "False" */

        int r2b = sb.size();
        sb.remove_at(i1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_size_post_c_161(ArrayList sa, ArrayList sb, int i1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
```

```
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        int r2a = sa.size();
        /*: assume "~(False)" */

        int r2b = sb.size();
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.remove_at(i1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void set_add_at_pre_s_162(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
            sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | (i1 > i2 & (ALL v. ((i1
            - 1, v) : sa..contents) = ((i1, v) : sa..contents)) & (i1 - 1, v1) :
            sa..contents & (i1, v1) : sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize &
            0 <= i1 & i1 < sa..csize)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        sb.add_at(i2, v2);
        Object r1b = sb.set(i1, v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void set_add_at_pre_c_162(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
            sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | (i1 > i2 & (ALL v. ((i1
            - 1, v) : sa..contents) = ((i1, v) : sa..contents)) & (i1 - 1, v1) :
            sa..contents & (i1, v1) : sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize &
            0 <= i1 & i1 < sa..csize))" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        /*: assume "0 <= i2 & i2 <= sb..csize" */
        sb.add_at(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.set(i1, v1);

```

```
5821        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
5822    }
5823
5824    static void set_add_at_between_s_163(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2, Object v2)
5825    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5826                  sa..contents = sb..contents & sa..csize = sb..csize"
5827        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
5828        ensures "True" */
5829    {
5830        /*: assume "0 <= i1 & i1 < sa..csize" */
5831        Object r1a = sa.set(i1, v1);
5832        /*: assume "i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | (i1 > i2 &
                (i1 - 1, r1a) : sa..contents & (i1 - 1, v1) : sa..contents & r1a = v1 & 0 <=
                i1 - 1 & i1 - 1 < sa..csize)" */
5833        /*: assume "0 <= i2 & i2 <= sa..csize" */
5834        sa.add_at(i2, v2);
5835
5836        sb.add_at(i2, v2);
5837        Object r1b = sb.set(i1, v1);
5838
5839        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
5840    }
5841
5842    static void set_add_at_between_c_163(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2, Object v2)
5843    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5844                  sa..contents = sb..contents & sa..csize = sb..csize"
5845        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
5846        ensures "True" */
5847    {
5848        /*: assume "0 <= i1 & i1 < sa..csize" */
5849        Object r1a = sa.set(i1, v1);
5850        /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | (i1 > i2 &
                (i1 - 1, r1a) : sa..contents & (i1 - 1, v1) : sa..contents & r1a = v1 & 0 <=
                i1 - 1 & i1 - 1 < sa..csize))" */
5851        /*: assume "0 <= i2 & i2 <= sa..csize" */
5852        sa.add_at(i2, v2);
5853
5854        /*: assume "0 <= i2 & i2 <= sb..csize" */
5855        sb.add_at(i2, v2);
5856        /*: assume "0 <= i1 & i1 < sb..csize" */
5857        Object r1b = sb.set(i1, v1);
5858
5859        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
5860    }
5861
5862    static void set_add_at_post_s_164(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2, Object v2)
5863    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5864                  sa..contents = sb..contents & sa..csize = sb..csize"
5865        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
5866        ensures "True" */
5867    {
5868        /*: assume "0 <= i1 & i1 < sa..csize" */
5869        Object r1a = sa.set(i1, v1);
5870        /*: assume "0 <= i2 & i2 <= sa..csize" */
5871        sa.add_at(i2, v2);
```

```
        /*: assume "i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | (i1 > i2 &
            (i1, r1a) : sa..contents & (i1, v1) : sa..contents & r1a = v1 & 0 <= i1 & i1
            < sa..csize)" */

        sb.add_at(i2, v2);
        Object r1b = sb.set(i1, v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void set_add_at_post_c_164(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);
        /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | (i1 > i2 &
            (i1, r1a) : sa..contents & (i1, v1) : sa..contents & r1a = v1 & 0 <= i1 & i1
            < sa..csize))" */

        /*: assume "0 <= i2 & i2 <= sb..csize" */
        sb.add_at(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.set(i1, v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void set_get_pre_s_165(ArrayList sa, ArrayList sb, int i1, Object v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & 0 <= i1 & i1 <
            sa..csize) | i1 > i2" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        Object r2b = sb.get(i2);
        Object r1b = sb.set(i1, v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void set_get_pre_c_165(ArrayList sa, ArrayList sb, int i1, Object v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & 0 <= i1 & i1 <
            sa..csize) | i1 > i2)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
```

```
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.get(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.set(i1, v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void set_get_between_s_166(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "i1 < i2 | (i1 = i2 & r1a = v1) | i1 > i2" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        Object r2b = sb.get(i2);
        Object r1b = sb.set(i1, v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void set_get_between_c_166(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v1) | i1 > i2)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.get(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.set(i1, v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void set_get_post_s_167(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);
```

```
5986        /*: assume "i1 < i2 | (i1 = i2 & r1a = v1) | i1 > i2" */
5987
5988        Object r2b = sb.get(i2);
5989        Object r1b = sb.set(i1, v1);
5990
5991        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
5992    }
5993
5994    static void set_get_post_c_167(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2)
5995    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5996                  sa..contents = sb..contents & sa..csize = sb..csize"
5997        modifies "sa..contents", "sb..contents"
5998        ensures "True" */
5999    {
6000        /*: assume "0 <= i1 & i1 < sa..csize" */
6001        Object r1a = sa.set(i1, v1);
6002        /*: assume "0 <= i2 & i2 < sa..csize" */
6003        Object r2a = sa.get(i2);
6004        /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v1) | i1 > i2)" */
6005
6006        /*: assume "0 <= i2 & i2 < sb..csize" */
6007        Object r2b = sb.get(i2);
6008        /*: assume "0 <= i1 & i1 < sb..csize" */
6009        Object r1b = sb.set(i1, v1);
6010
6011        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
6012    }
6013
6014    static void set_indexOf_pre_s_168(ArrayList sa, ArrayList sb, int i1, Object v1,
            Object v2)
6015    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6016                  sa..contents = sb..contents & sa..csize = sb..csize"
6017        modifies "sa..contents", "sb..contents"
6018        ensures "True" */
6019    {
6020        /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
                v2) | (EX i. (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) :
                sa..contents & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 <
                sa..csize & v1 = v2) | (~(EX i. (i, v2) : sa..contents & 0 <= i & i <= i1) &
                (EX i. (i, v2) : sa..contents & i1 < i & i < sa..csize) & v1 ~= v2)" */
6021        /*: assume "0 <= i1 & i1 < sa..csize" */
6022        Object r1a = sa.set(i1, v1);
6023        int r2a = sa.indexOf(v2);
6024
6025        int r2b = sb.indexOf(v2);
6026        Object r1b = sb.set(i1, v1);
6027
6028        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
6029    }
6030
6031    static void set_indexOf_pre_c_168(ArrayList sa, ArrayList sb, int i1, Object v1,
            Object v2)
6032    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6033                  sa..contents = sb..contents & sa..csize = sb..csize"
6034        modifies "sa..contents", "sb..contents"
6035        ensures "True" */
6036    {
6037        /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
                v2) | (EX i. (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) :
                sa..contents & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 <
```

```
            sa..csize & v1 = v2) | (~(EX i. (i, v2) : sa..contents & 0 <= i & i <= i1) &
            (EX i. (i, v2) : sa..contents & i1 < i & i < sa..csize) & v1 ~= v2))" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        int r2a = sa.indexOf(v2);

        int r2b = sb.indexOf(v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.set(i1, v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void set_indexOf_between_s_169(ArrayList sa, ArrayList sb, int i1, Object
        v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & r1a ~=
            v2) | (EX i. (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) :
            sa..contents & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 <
            sa..csize & r1a = v2) | (~(EX i. (i, v2) : sa..contents & 0 <= i & i <= i1)
            & (EX i. (i, v2) : sa..contents & i1 < i & i < sa..csize) & r1a ~= v2)" */
        int r2a = sa.indexOf(v2);

        int r2b = sb.indexOf(v2);
        Object r1b = sb.set(i1, v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void set_indexOf_between_c_169(ArrayList sa, ArrayList sb, int i1, Object
        v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & r1a
            ~= v2) | (EX i. (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2)
            : sa..contents & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 <
            sa..csize & r1a = v2) | (~(EX i. (i, v2) : sa..contents & 0 <= i & i <= i1)
            & (EX i. (i, v2) : sa..contents & i1 < i & i < sa..csize) & r1a ~= v2))" */
        int r2a = sa.indexOf(v2);

        int r2b = sb.indexOf(v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.set(i1, v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void set_indexOf_post_s_170(ArrayList sa, ArrayList sb, int i1, Object v1,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
```

```
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        int r2a = sa.indexOf(v2);
        /*: assume "(r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1) | (r2a = i1 & r1a =
            v2) | (r2a > i1 & r1a ~= v2)" */

        int r2b = sb.indexOf(v2);
        Object r1b = sb.set(i1, v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void set_indexOf_post_c_170(ArrayList sa, ArrayList sb, int i1, Object v1,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        int r2a = sa.indexOf(v2);
        /*: assume "~((r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1) | (r2a = i1 & r1a =
            v2) | (r2a > i1 & r1a ~= v2))" */

        int r2b = sb.indexOf(v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.set(i1, v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void set_lastIndexOf_pre_s_171(ArrayList sa, ArrayList sb, int i1, Object
        v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
            v2) | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
            sa..contents & i1 <= i & i < sa..csize) & v1 ~= v2) | ((i1, v2) :
            sa..contents & 0 <= i1 & i1 < sa..csize & ~(EX i. (i, v2) : sa..contents &
            i1 < i & i < sa..csize) & v1 = v2) | (EX i. (i, v2) : sa..contents & i1 < i
            & i < sa..csize)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        int r2a = sa.lastIndexOf(v2);

        int r2b = sb.lastIndexOf(v2);
        Object r1b = sb.set(i1, v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void set_lastIndexOf_pre_c_171(ArrayList sa, ArrayList sb, int i1, Object
        v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
```

```
6139        modifies "sa..contents", "sb..contents"
6140        ensures "True" */
6141    {
6142        /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
                v2) | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
                sa..contents & i1 <= i & i < sa..csize) & v1 ~= v2) | ((i1, v2) :
                sa..contents & 0 <= i1 & i1 < sa..csize & ~(EX i. (i, v2) : sa..contents &
                i1 < i & i < sa..csize) & v1 = v2) | (EX i. (i, v2) : sa..contents & i1 < i
                & i < sa..csize))" */
6143        /*: assume "0 <= i1 & i1 < sa..csize" */
6144        Object r1a = sa.set(i1, v1);
6145        int r2a = sa.lastIndexOf(v2);
6146
6147        int r2b = sb.lastIndexOf(v2);
6148        /*: assume "0 <= i1 & i1 < sb..csize" */
6149        Object r1b = sb.set(i1, v1);
6150
6151        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
6152    }
6153
6154    static void set_lastIndexOf_between_s_172(ArrayList sa, ArrayList sb, int i1,
            Object v1, Object v2)
6155    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6156                  sa..contents = sb..contents & sa..csize = sb..csize"
6157        modifies "sa..contents", "sb..contents"
6158        ensures "True" */
6159    {
6160        /*: assume "0 <= i1 & i1 < sa..csize" */
6161        Object r1a = sa.set(i1, v1);
6162        /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & r1a ~=
                v2) | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
                sa..contents & i1 <= i & i < sa..csize) & r1a ~= v2) | ((i1, v2) :
                sa..contents & 0 <= i1 & i1 < sa..csize & ~(EX i. (i, v2) : sa..contents &
                i1 < i & i < sa..csize) & r1a = v2) | (EX i. (i, v2) : sa..contents & i1 < i
                & i < sa..csize)" */
6163        int r2a = sa.lastIndexOf(v2);
6164
6165        int r2b = sb.lastIndexOf(v2);
6166        Object r1b = sb.set(i1, v1);
6167
6168        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
6169    }
6170
6171    static void set_lastIndexOf_between_c_172(ArrayList sa, ArrayList sb, int i1,
            Object v1, Object v2)
6172    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6173                  sa..contents = sb..contents & sa..csize = sb..csize"
6174        modifies "sa..contents", "sb..contents"
6175        ensures "True" */
6176    {
6177        /*: assume "0 <= i1 & i1 < sa..csize" */
6178        Object r1a = sa.set(i1, v1);
6179        /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & r1a
                ~= v2) | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2)
                : sa..contents & i1 <= i & i < sa..csize) & r1a ~= v2) | ((i1, v2) :
                sa..contents & 0 <= i1 & i1 < sa..csize & ~(EX i. (i, v2) : sa..contents &
                i1 < i & i < sa..csize) & r1a = v2) | (EX i. (i, v2) : sa..contents & i1 < i
                & i < sa..csize))" */
6180        int r2a = sa.lastIndexOf(v2);
6181
6182        int r2b = sb.lastIndexOf(v2);
6183        /*: assume "0 <= i1 & i1 < sb..csize" */
6184        Object r1b = sb.set(i1, v1);
```

```
6185
6186        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
6187    }
6188
6189    static void set_lastIndexOf_post_s_173(ArrayList sa, ArrayList sb, int i1, Object
            v1, Object v2)
6190    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6191                  sa..contents = sb..contents & sa..csize = sb..csize"
6192        modifies "sa..contents", "sb..contents"
6193        ensures "True" */
6194    {
6195        /*: assume "0 <= i1 & i1 < sa..csize" */
6196        Object r1a = sa.set(i1, v1);
6197        int r2a = sa.lastIndexOf(v2);
6198        /*: assume "(r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1 & r1a ~= v2) | (r2a =
                i1 & r1a = v2) | r2a > i1" */
6199
6200        int r2b = sb.lastIndexOf(v2);
6201        Object r1b = sb.set(i1, v1);
6202
6203        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
6204    }
6205
6206    static void set_lastIndexOf_post_c_173(ArrayList sa, ArrayList sb, int i1, Object
            v1, Object v2)
6207    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6208                  sa..contents = sb..contents & sa..csize = sb..csize"
6209        modifies "sa..contents", "sb..contents"
6210        ensures "True" */
6211    {
6212        /*: assume "0 <= i1 & i1 < sa..csize" */
6213        Object r1a = sa.set(i1, v1);
6214        int r2a = sa.lastIndexOf(v2);
6215        /*: assume "~((r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1 & r1a ~= v2) | (r2a
                = i1 & r1a = v2) | r2a > i1)" */
6216
6217        int r2b = sb.lastIndexOf(v2);
6218        /*: assume "0 <= i1 & i1 < sb..csize" */
6219        Object r1b = sb.set(i1, v1);
6220
6221        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
6222    }
6223
6224    static void set_remove_at_pre_s_174(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2)
6225    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6226                  sa..contents = sb..contents & sa..csize = sb..csize"
6227        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
6228        ensures "True" */
6229    {
6230        /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (ALL v. ((i1, v) :
                sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
                (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
                1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
                sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
                (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
                1 < sa..csize)" */
6231        /*: assume "0 <= i1 & i1 < sa..csize" */
6232        Object r1a = sa.set(i1, v1);
6233        /*: assume "0 <= i2 & i2 < sa..csize" */
6234        Object r2a = sa.remove_at(i2);
6235
```

```
        Object r2b = sb.remove_at(i2);
        Object r1b = sb.set(i1, v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void set_remove_at_pre_c_174(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (ALL v. ((i1, v) :
            sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
            (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
            1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
            sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
            (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
            1 < sa..csize))" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.set(i1, v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void set_remove_at_between_s_175(ArrayList sa, ArrayList sb, int i1, Object
        v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, r1a) :
            sa..contents & r1a = v1 & (i1 + 1, v1) : sa..contents & 0 <= i1 + 1 & i1 + 1
            < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (i1 + 1, r1a) : sa..contents
            & r1a = v1 & (i1 + 1, v1) : sa..contents & 0 <= i1 + 1 & i1 + 1 <
            sa..csize)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);

        Object r2b = sb.remove_at(i2);
        Object r1b = sb.set(i1, v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void set_remove_at_between_c_175(ArrayList sa, ArrayList sb, int i1, Object
        v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
```

```
6285    {
6286        /*: assume "0 <= i1 & i1 < sa..csize" */
6287        Object r1a = sa.set(i1, v1);
6288        /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, r1a) :
                sa..contents & r1a = v1 & (i1 + 1, v1) : sa..contents & 0 <= i1 + 1 & i1 + 1
                < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (i1 + 1, r1a) : sa..contents
                & r1a = v1 & (i1 + 1, v1) : sa..contents & 0 <= i1 + 1 & i1 + 1 <
                sa..csize))" */
6289        /*: assume "0 <= i2 & i2 < sa..csize" */
6290        Object r2a = sa.remove_at(i2);
6291
6292        /*: assume "0 <= i2 & i2 < sb..csize" */
6293        Object r2b = sb.remove_at(i2);
6294        /*: assume "0 <= i1 & i1 < sb..csize" */
6295        Object r1b = sb.set(i1, v1);
6296
6297        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
6298    }
6299
6300    static void set_remove_at_post_s_176(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2)
6301    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6302                  sa..contents = sb..contents & sa..csize = sb..csize"
6303        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
6304        ensures "True" */
6305    {
6306        /*: assume "0 <= i1 & i1 < sa..csize" */
6307        Object r1a = sa.set(i1, v1);
6308        /*: assume "0 <= i2 & i2 < sa..csize" */
6309        Object r2a = sa.remove_at(i2);
6310        /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize & (i1, r1a) : sa..contents &
                r1a = v1 & (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize) | (sa..csize
                > i1 & i1 > i2 & (i1, r1a) : sa..contents & r1a = v1 & (i1, v1) :
                sa..contents & 0 <= i1 & i1 < sa..csize)" */
6311
6312        Object r2b = sb.remove_at(i2);
6313        Object r1b = sb.set(i1, v1);
6314
6315        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
6316    }
6317
6318    static void set_remove_at_post_c_176(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2)
6319    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6320                  sa..contents = sb..contents & sa..csize = sb..csize"
6321        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
6322        ensures "True" */
6323    {
6324        /*: assume "0 <= i1 & i1 < sa..csize" */
6325        Object r1a = sa.set(i1, v1);
6326        /*: assume "0 <= i2 & i2 < sa..csize" */
6327        Object r2a = sa.remove_at(i2);
6328        /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize & (i1, r1a) : sa..contents &
                r1a = v1 & (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize) | (sa..csize
                > i1 & i1 > i2 & (i1, r1a) : sa..contents & r1a = v1 & (i1, v1) :
                sa..contents & 0 <= i1 & i1 < sa..csize))" */
6329
6330        /*: assume "0 <= i2 & i2 < sb..csize" */
6331        Object r2b = sb.remove_at(i2);
6332        /*: assume "0 <= i1 & i1 < sb..csize" */
6333        Object r1b = sb.set(i1, v1);
6334
```

```
6335        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
6336    }
6337
6338    static void set_remove_at_pre_s_177(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2)
6339    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6340                  sa..contents = sb..contents & sa..csize = sb..csize"
6341        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
6342        ensures "True" */
6343    {
6344        /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (ALL v. ((i1, v) :
                sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
                (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
                1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
                sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
                (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
                1 < sa..csize)" */
6345        /*: assume "0 <= i1 & i1 < sa..csize" */
6346        Object r1a = sa.set(i1, v1);
6347        /*: assume "0 <= i2 & i2 < sa..csize" */
6348        sa.remove_at(i2);
6349
6350        sb.remove_at(i2);
6351        Object r1b = sb.set(i1, v1);
6352
6353        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
6354    }
6355
6356    static void set_remove_at_pre_c_177(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2)
6357    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6358                  sa..contents = sb..contents & sa..csize = sb..csize"
6359        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
6360        ensures "True" */
6361    {
6362        /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (ALL v. ((i1, v) :
                sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
                (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
                1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
                sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
                (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
                1 < sa..csize))" */
6363        /*: assume "0 <= i1 & i1 < sa..csize" */
6364        Object r1a = sa.set(i1, v1);
6365        /*: assume "0 <= i2 & i2 < sa..csize" */
6366        sa.remove_at(i2);
6367
6368        /*: assume "0 <= i2 & i2 < sb..csize" */
6369        sb.remove_at(i2);
6370        /*: assume "0 <= i1 & i1 < sb..csize" */
6371        Object r1b = sb.set(i1, v1);
6372
6373        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
6374    }
6375
6376    static void set_remove_at_between_s_178(ArrayList sa, ArrayList sb, int i1, Object
            v1, int i2)
6377    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6378                  sa..contents = sb..contents & sa..csize = sb..csize"
6379        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
6380        ensures "True" */
6381    {
6382        /*: assume "0 <= i1 & i1 < sa..csize" */
```

```
6383        Object r1a = sa.set(i1, v1);
6384        /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, r1a) :
                sa..contents & r1a = v1 & (i1 + 1, v1) : sa..contents & 0 <= i1 + 1 & i1 + 1
                < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (i1 + 1, r1a) : sa..contents
                & r1a = v1 & (i1 + 1, v1) : sa..contents & 0 <= i1 + 1 & i1 + 1 <
                sa..csize)" */
6385        /*: assume "0 <= i2 & i2 < sa..csize" */
6386        sa.remove_at(i2);
6387
6388        sb.remove_at(i2);
6389        Object r1b = sb.set(i1, v1);
6390
6391        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
6392    }
6393
6394    static void set_remove_at_between_c_178(ArrayList sa, ArrayList sb, int i1, Object
            v1, int i2)
6395    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6396                  sa..contents = sb..contents & sa..csize = sb..csize"
6397        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
6398        ensures "True" */
6399    {
6400        /*: assume "0 <= i1 & i1 < sa..csize" */
6401        Object r1a = sa.set(i1, v1);
6402        /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, r1a) :
                sa..contents & r1a = v1 & (i1 + 1, v1) : sa..contents & 0 <= i1 + 1 & i1 + 1
                < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (i1 + 1, r1a) : sa..contents
                & r1a = v1 & (i1 + 1, v1) : sa..contents & 0 <= i1 + 1 & i1 + 1 <
                sa..csize))" */
6403        /*: assume "0 <= i2 & i2 < sa..csize" */
6404        sa.remove_at(i2);
6405
6406        /*: assume "0 <= i2 & i2 < sb..csize" */
6407        sb.remove_at(i2);
6408        /*: assume "0 <= i1 & i1 < sb..csize" */
6409        Object r1b = sb.set(i1, v1);
6410
6411        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
6412    }
6413
6414    static void set_remove_at_post_s_179(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2)
6415    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6416                  sa..contents = sb..contents & sa..csize = sb..csize"
6417        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
6418        ensures "True" */
6419    {
6420        /*: assume "0 <= i1 & i1 < sa..csize" */
6421        Object r1a = sa.set(i1, v1);
6422        /*: assume "0 <= i2 & i2 < sa..csize" */
6423        sa.remove_at(i2);
6424        /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize & (i1, r1a) : sa..contents &
                r1a = v1 & (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize) | (sa..csize
                > i1 & i1 > i2 & (i1, r1a) : sa..contents & r1a = v1 & (i1, v1) :
                sa..contents & 0 <= i1 & i1 < sa..csize)" */
6425
6426        sb.remove_at(i2);
6427        Object r1b = sb.set(i1, v1);
6428
6429        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
6430    }
6431
6432    static void set_remove_at_post_c_179(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2)
```

```
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);
        /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize & (i1, r1a) : sa..contents &
            r1a = v1 & (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize) | (sa..csize
            > i1 & i1 > i2 & (i1, r1a) : sa..contents & r1a = v1 & (i1, v1) :
            sa..contents & 0 <= i1 & i1 < sa..csize))" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.set(i1, v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void set_set_pre_s_180(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
            sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | i1 > i2" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        Object r2b = sb.set(i2, v2);
        Object r1b = sb.set(i1, v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void set_set_pre_c_180(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
            sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | i1 > i2)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.set(i1, v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
```

```
6488     }
6489
6490     static void set_set_between_s_181(ArrayList sa, ArrayList sb, int i1, Object v1,
             int i2, Object v2)
6491     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6492                 sa..contents = sb..contents & sa..csize = sb..csize"
6493         modifies "sa..contents", "sb..contents"
6494         ensures "True" */
6495     {
6496         /*: assume "0 <= i1 & i1 < sa..csize" */
6497         Object r1a = sa.set(i1, v1);
6498         /*: assume "i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | i1 > i2" */
6499         /*: assume "0 <= i2 & i2 < sa..csize" */
6500         Object r2a = sa.set(i2, v2);
6501
6502         Object r2b = sb.set(i2, v2);
6503         Object r1b = sb.set(i1, v1);
6504
6505         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                 sb..csize" */
6506     }
6507
6508     static void set_set_between_c_181(ArrayList sa, ArrayList sb, int i1, Object v1,
             int i2, Object v2)
6509     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6510                 sa..contents = sb..contents & sa..csize = sb..csize"
6511         modifies "sa..contents", "sb..contents"
6512         ensures "True" */
6513     {
6514         /*: assume "0 <= i1 & i1 < sa..csize" */
6515         Object r1a = sa.set(i1, v1);
6516         /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | i1 > i2)" */
6517         /*: assume "0 <= i2 & i2 < sa..csize" */
6518         Object r2a = sa.set(i2, v2);
6519
6520         /*: assume "0 <= i2 & i2 < sb..csize" */
6521         Object r2b = sb.set(i2, v2);
6522         /*: assume "0 <= i1 & i1 < sb..csize" */
6523         Object r1b = sb.set(i1, v1);
6524
6525         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                 sb..csize)" */
6526     }
6527
6528     static void set_set_post_s_182(ArrayList sa, ArrayList sb, int i1, Object v1, int
             i2, Object v2)
6529     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6530                 sa..contents = sb..contents & sa..csize = sb..csize"
6531         modifies "sa..contents", "sb..contents"
6532         ensures "True" */
6533     {
6534         /*: assume "0 <= i1 & i1 < sa..csize" */
6535         Object r1a = sa.set(i1, v1);
6536         /*: assume "0 <= i2 & i2 < sa..csize" */
6537         Object r2a = sa.set(i2, v2);
6538         /*: assume "i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | i1 > i2" */
6539
6540         Object r2b = sb.set(i2, v2);
6541         Object r1b = sb.set(i1, v1);
6542
6543         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                 sb..csize" */
6544     }
6545
```

```
    static void set_set_post_c_182(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);
        /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | i1 > i2)" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.set(i1, v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void set_set_pre_s_183(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
            sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | i1 > i2" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        sb.set(i2, v2);
        Object r1b = sb.set(i1, v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void set_set_pre_c_183(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
            sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | i1 > i2)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.set(i1, v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }
```

```
6604    static void set_set_between_s_184(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2, Object v2)
6605    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6606                sa..contents = sb..contents & sa..csize = sb..csize"
6607        modifies "sa..contents", "sb..contents"
6608        ensures "True" */
6609    {
6610        /*: assume "0 <= i1 & i1 < sa..csize" */
6611        Object r1a = sa.set(i1, v1);
6612        /*: assume "i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | i1 > i2" */
6613        /*: assume "0 <= i2 & i2 < sa..csize" */
6614        sa.set(i2, v2);
6615
6616        sb.set(i2, v2);
6617        Object r1b = sb.set(i1, v1);
6618
6619        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
6620    }
6621
6622    static void set_set_between_c_184(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2, Object v2)
6623    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6624                sa..contents = sb..contents & sa..csize = sb..csize"
6625        modifies "sa..contents", "sb..contents"
6626        ensures "True" */
6627    {
6628        /*: assume "0 <= i1 & i1 < sa..csize" */
6629        Object r1a = sa.set(i1, v1);
6630        /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | i1 > i2)" */
6631        /*: assume "0 <= i2 & i2 < sa..csize" */
6632        sa.set(i2, v2);
6633
6634        /*: assume "0 <= i2 & i2 < sb..csize" */
6635        sb.set(i2, v2);
6636        /*: assume "0 <= i1 & i1 < sb..csize" */
6637        Object r1b = sb.set(i1, v1);
6638
6639        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
6640    }
6641
6642    static void set_set_post_s_185(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2, Object v2)
6643    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6644                sa..contents = sb..contents & sa..csize = sb..csize"
6645        modifies "sa..contents", "sb..contents"
6646        ensures "True" */
6647    {
6648        /*: assume "0 <= i1 & i1 < sa..csize" */
6649        Object r1a = sa.set(i1, v1);
6650        /*: assume "0 <= i2 & i2 < sa..csize" */
6651        sa.set(i2, v2);
6652        /*: assume "i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | i1 > i2" */
6653
6654        sb.set(i2, v2);
6655        Object r1b = sb.set(i1, v1);
6656
6657        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
6658    }
6659
6660    static void set_set_post_c_185(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2, Object v2)
6661    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6662                sa..contents = sb..contents & sa..csize = sb..csize"
6663        modifies "sa..contents", "sb..contents"
```

```
6664        ensures "True" */
6665    {
6666        /*: assume "0 <= i1 & i1 < sa..csize" */
6667        Object r1a = sa.set(i1, v1);
6668        /*: assume "0 <= i2 & i2 < sa..csize" */
6669        sa.set(i2, v2);
6670        /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | i1 > i2)" */
6671
6672        /*: assume "0 <= i2 & i2 < sb..csize" */
6673        sb.set(i2, v2);
6674        /*: assume "0 <= i1 & i1 < sb..csize" */
6675        Object r1b = sb.set(i1, v1);
6676
6677        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
6678    }
6679
6680    static void set_size_pre_s_186(ArrayList sa, ArrayList sb, int i1, Object v1)
6681    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6682                  sa..contents = sb..contents & sa..csize = sb..csize"
6683        modifies "sa..contents", "sb..contents"
6684        ensures "True" */
6685    {
6686        /*: assume "True" */
6687        /*: assume "0 <= i1 & i1 < sa..csize" */
6688        Object r1a = sa.set(i1, v1);
6689        int r2a = sa.size();
6690
6691        int r2b = sb.size();
6692        Object r1b = sb.set(i1, v1);
6693
6694        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
6695    }
6696
6697    static void set_size_pre_c_186(ArrayList sa, ArrayList sb, int i1, Object v1)
6698    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6699                  sa..contents = sb..contents & sa..csize = sb..csize"
6700        modifies "sa..contents", "sb..contents"
6701        ensures "True" */
6702    {
6703        /*: assume "~(True)" */
6704        /*: assume "0 <= i1 & i1 < sa..csize" */
6705        Object r1a = sa.set(i1, v1);
6706        int r2a = sa.size();
6707
6708        int r2b = sb.size();
6709        /*: assume "0 <= i1 & i1 < sb..csize" */
6710        Object r1b = sb.set(i1, v1);
6711
6712        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
6713    }
6714
6715    static void set_size_between_s_187(ArrayList sa, ArrayList sb, int i1, Object v1)
6716    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6717                  sa..contents = sb..contents & sa..csize = sb..csize"
6718        modifies "sa..contents", "sb..contents"
6719        ensures "True" */
6720    {
6721        /*: assume "0 <= i1 & i1 < sa..csize" */
6722        Object r1a = sa.set(i1, v1);
6723        /*: assume "True" */
6724        int r2a = sa.size();
6725
```

```
6726        int r2b = sb.size();
6727        Object r1b = sb.set(i1, v1);
6728
6729        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
6730    }
6731
6732    static void set_size_between_c_187(ArrayList sa, ArrayList sb, int i1, Object v1)
6733    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6734                  sa..contents = sb..contents & sa..csize = sb..csize"
6735        modifies "sa..contents", "sb..contents"
6736        ensures "True" */
6737    {
6738        /*: assume "0 <= i1 & i1 < sa..csize" */
6739        Object r1a = sa.set(i1, v1);
6740        /*: assume "~(True)" */
6741        int r2a = sa.size();
6742
6743        int r2b = sb.size();
6744        /*: assume "0 <= i1 & i1 < sb..csize" */
6745        Object r1b = sb.set(i1, v1);
6746
6747        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
6748    }
6749
6750    static void set_size_post_s_188(ArrayList sa, ArrayList sb, int i1, Object v1)
6751    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6752                  sa..contents = sb..contents & sa..csize = sb..csize"
6753        modifies "sa..contents", "sb..contents"
6754        ensures "True" */
6755    {
6756        /*: assume "0 <= i1 & i1 < sa..csize" */
6757        Object r1a = sa.set(i1, v1);
6758        int r2a = sa.size();
6759        /*: assume "True" */
6760
6761        int r2b = sb.size();
6762        Object r1b = sb.set(i1, v1);
6763
6764        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
6765    }
6766
6767    static void set_size_post_c_188(ArrayList sa, ArrayList sb, int i1, Object v1)
6768    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6769                  sa..contents = sb..contents & sa..csize = sb..csize"
6770        modifies "sa..contents", "sb..contents"
6771        ensures "True" */
6772    {
6773        /*: assume "0 <= i1 & i1 < sa..csize" */
6774        Object r1a = sa.set(i1, v1);
6775        int r2a = sa.size();
6776        /*: assume "~(True)" */
6777
6778        int r2b = sb.size();
6779        /*: assume "0 <= i1 & i1 < sb..csize" */
6780        Object r1b = sb.set(i1, v1);
6781
6782        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
6783    }
6784
6785    static void set_add_at_pre_s_189(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
```

```
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
            sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | (i1 > i2 & (ALL v. ((i1
            - 1, v) : sa..contents) = ((i1, v) : sa..contents)) & (i1 - 1, v1) :
            sa..contents & (i1, v1) : sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize &
            0 <= i1 & i1 < sa..csize)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        sb.add_at(i2, v2);
        sb.set(i1, v1);

        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void set_add_at_pre_c_189(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
            sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | (i1 > i2 & (ALL v. ((i1
            - 1, v) : sa..contents) = ((i1, v) : sa..contents)) & (i1 - 1, v1) :
            sa..contents & (i1, v1) : sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize &
            0 <= i1 & i1 < sa..csize))" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        /*: assume "0 <= i2 & i2 <= sb..csize" */
        sb.add_at(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.set(i1, v1);

        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
    }

    static void set_add_at_between_s_190(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.set(i1, v1);
        /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & (i1, v2) :
            sa..(old contents) & v1 = v2 & 0 <= i1 & i1 < sa..(old csize)) | (i1 > i2 &
            (ALL v. ((i1 - 1, v) : sa..contents) = ((i1, v) : sa..(old contents))) & (i1
            - 1, v1) : sa..contents & (i1, v1) : sa..(old contents) & 0 <= i1 - 1 & i1 -
            1 < sa..csize & 0 <= i1 & i1 < sa..(old csize))" */
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);
```

```
6834
6835        sb.add_at(i2, v2);
6836        sb.set(i1, v1);
6837
6838        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
6839    }
6840
6841    static void set_add_at_between_c_190(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2, Object v2)
6842    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6843                sa..contents = sb..contents & sa..csize = sb..csize"
6844        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
6845        ensures "True" */
6846    {
6847        /*: assume "0 <= i1 & i1 < sa..csize" */
6848        sa.set(i1, v1);
6849        /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & (i1, v2) :
            sa..(old contents) & v1 = v2 & 0 <= i1 & i1 < sa..(old csize)) | (i1 > i2 &
            (ALL v. ((i1 - 1, v) : sa..contents) = ((i1, v) : sa..(old contents))) & (i1
            - 1, v1) : sa..contents & (i1, v1) : sa..(old contents) & 0 <= i1 - 1 & i1 -
            1 < sa..csize & 0 <= i1 & i1 < sa..(old csize)))" */
6850        /*: assume "0 <= i2 & i2 <= sa..csize" */
6851        sa.add_at(i2, v2);
6852
6853        /*: assume "0 <= i2 & i2 <= sb..csize" */
6854        sb.add_at(i2, v2);
6855        /*: assume "0 <= i1 & i1 < sb..csize" */
6856        sb.set(i1, v1);
6857
6858        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
6859    }
6860
6861    static void set_add_at_post_s_191(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2, Object v2)
6862    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6863                sa..contents = sb..contents & sa..csize = sb..csize"
6864        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
6865        ensures "True" */
6866    {
6867        /*: assume "0 <= i1 & i1 < sa..csize" */
6868        sa.set(i1, v1);
6869        /*: assume "0 <= i2 & i2 <= sa..csize" */
6870        sa.add_at(i2, v2);
6871        /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & (i1, v2) :
            sa..(old contents) & v1 = v2 & 0 <= i1 & i1 < sa..(old csize)) | (i1 > i2 &
            (ALL v. ((i1, v) : sa..contents) = ((i1, v) : sa..(old contents))) & (i1, v1)
            : sa..contents & (i1, v1) : sa..(old contents) & 0 <= i1 & i1 < sa..csize &
            0 <= i1 & i1 < sa..(old csize)))" */
6872
6873        sb.add_at(i2, v2);
6874        sb.set(i1, v1);
6875
6876        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
6877    }
6878
6879    static void set_add_at_post_c_191(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2, Object v2)
6880    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6881                sa..contents = sb..contents & sa..csize = sb..csize"
6882        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
6883        ensures "True" */
6884    {
```

```
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);
        /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & (i1, v2) :
            sa..(old contents) & v1 = v2 & 0 <= i1 & i1 < sa..(old csize)) | (i1 > i2 &
            (ALL v. ((i1, v) : sa..contents) = ((i1, v) : sa..(old contents)) & (i1, v1)
            : sa..contents & (i1, v1) : sa..(old contents) & 0 <= i1 & i1 < sa..csize &
            0 <= i1 & i1 < sa..(old csize))))" */

        /*: assume "0 <= i2 & i2 <= sb..csize" */
        sb.add_at(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.set(i1, v1);

        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
    }

    static void set_get_pre_s_192(ArrayList sa, ArrayList sb, int i1, Object v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & 0 <= i1 & i1 <
            sa..csize) | i1 > i2" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        Object r2b = sb.get(i2);
        sb.set(i1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void set_get_pre_c_192(ArrayList sa, ArrayList sb, int i1, Object v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & 0 <= i1 & i1 <
            sa..csize) | i1 > i2)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.get(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.set(i1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void set_get_between_s_193(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
```

```
6942    {
6943        /*: assume "0 <= i1 & i1 < sa..csize" */
6944        sa.set(i1, v1);
6945        /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & 0 <= i1 & i1 <
                sa..(old csize)) | i1 > i2" */
6946        /*: assume "0 <= i2 & i2 < sa..csize" */
6947        Object r2a = sa.get(i2);
6948
6949        Object r2b = sb.get(i2);
6950        sb.set(i1, v1);
6951
6952        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
6953    }
6954
6955    static void set_get_between_c_193(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2)
6956    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6957                  sa..contents = sb..contents & sa..csize = sb..csize"
6958        modifies "sa..contents", "sb..contents"
6959        ensures "True" */
6960    {
6961        /*: assume "0 <= i1 & i1 < sa..csize" */
6962        sa.set(i1, v1);
6963        /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & 0 <= i1 & i1
                < sa..(old csize)) | i1 > i2)" */
6964        /*: assume "0 <= i2 & i2 < sa..csize" */
6965        Object r2a = sa.get(i2);
6966
6967        /*: assume "0 <= i2 & i2 < sb..csize" */
6968        Object r2b = sb.get(i2);
6969        /*: assume "0 <= i1 & i1 < sb..csize" */
6970        sb.set(i1, v1);
6971
6972        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
6973    }
6974
6975    static void set_get_post_s_194(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2)
6976    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6977                  sa..contents = sb..contents & sa..csize = sb..csize"
6978        modifies "sa..contents", "sb..contents"
6979        ensures "True" */
6980    {
6981        /*: assume "0 <= i1 & i1 < sa..csize" */
6982        sa.set(i1, v1);
6983        /*: assume "0 <= i2 & i2 < sa..csize" */
6984        Object r2a = sa.get(i2);
6985        /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & 0 <= i1 & i1 <
                sa..(old csize)) | i1 > i2" */
6986
6987        Object r2b = sb.get(i2);
6988        sb.set(i1, v1);
6989
6990        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
6991    }
6992
6993    static void set_get_post_c_194(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2)
6994    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6995                  sa..contents = sb..contents & sa..csize = sb..csize"
6996        modifies "sa..contents", "sb..contents"
6997        ensures "True" */
6998    {
6999        /*: assume "0 <= i1 & i1 < sa..csize" */
```

```
        sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);
        /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & 0 <= i1 & i1
            < sa..(old csize)) | i1 > i2)" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.get(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.set(i1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void set_indexOf_pre_s_195(ArrayList sa, ArrayList sb, int i1, Object v1,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
            v2) | (EX i. (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) :
            sa..contents & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 <
            sa..csize & v1 = v2) | (~(EX i. (i, v2) : sa..contents & 0 <= i & i <= i1) &
            (EX i. (i, v2) : sa..contents & i1 < i & i < sa..csize) & v1 ~= v2)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.set(i1, v1);
        int r2a = sa.indexOf(v2);

        int r2b = sb.indexOf(v2);
        sb.set(i1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void set_indexOf_pre_c_195(ArrayList sa, ArrayList sb, int i1, Object v1,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
            v2) | (EX i. (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) :
            sa..contents & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 <
            sa..csize & v1 = v2) | (~(EX i. (i, v2) : sa..contents & 0 <= i & i <= i1) &
            (EX i. (i, v2) : sa..contents & i1 < i & i < sa..csize) & v1 ~= v2))" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.set(i1, v1);
        int r2a = sa.indexOf(v2);

        int r2b = sb.indexOf(v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.set(i1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void set_indexOf_between_s_196(ArrayList sa, ArrayList sb, int i1, Object
        v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
```

```
7051        modifies "sa..contents", "sb..contents"
7052        ensures "True" */
7053    {
7054        /*: assume "0 <= i1 & i1 < sa..csize" */
7055        sa.set(i1, v1);
7056        /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & (i1,
                v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | (EX i. (i, v2)
                : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) : sa..contents & 0 <= i
                & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & (i1, v2) :
                sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | (~(EX i. (i, v2) :
                sa..contents & 0 <= i & i <= i1) & (EX i. (i, v2) : sa..contents & i1 < i &
                i < sa..csize) & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old
                csize))" */
7057        int r2a = sa.indexOf(v2);
7058
7059        int r2b = sb.indexOf(v2);
7060        sb.set(i1, v1);
7061
7062        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
7063    }
7064
7065    static void set_indexOf_between_c_196(ArrayList sa, ArrayList sb, int i1, Object
            v1, Object v2)
7066    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7067                  sa..contents = sb..contents & sa..csize = sb..csize"
7068        modifies "sa..contents", "sb..contents"
7069        ensures "True" */
7070    {
7071        /*: assume "0 <= i1 & i1 < sa..csize" */
7072        sa.set(i1, v1);
7073        /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & (i1,
                v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | (EX i. (i, v2)
                : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) : sa..contents & 0 <= i
                & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & (i1, v2) :
                sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | (~(EX i. (i, v2) :
                sa..contents & 0 <= i & i <= i1) & (EX i. (i, v2) : sa..contents & i1 < i &
                i < sa..csize) & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old
                csize)))" */
7074        int r2a = sa.indexOf(v2);
7075
7076        int r2b = sb.indexOf(v2);
7077        /*: assume "0 <= i1 & i1 < sb..csize" */
7078        sb.set(i1, v1);
7079
7080        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
7081    }
7082
7083    static void set_indexOf_post_s_197(ArrayList sa, ArrayList sb, int i1, Object v1,
            Object v2)
7084    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7085                  sa..contents = sb..contents & sa..csize = sb..csize"
7086        modifies "sa..contents", "sb..contents"
7087        ensures "True" */
7088    {
7089        /*: assume "0 <= i1 & i1 < sa..csize" */
7090        sa.set(i1, v1);
7091        int r2a = sa.indexOf(v2);
7092        /*: assume "(r2a < 0 & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old
                csize)) | (0 <= r2a & r2a < i1) | (r2a = i1 & (i1, v2) : sa..(old contents)
                & 0 <= i1 & i1 < sa..(old csize)) | (r2a > i1 & (i1, v2) ~: sa..(old
                contents) & 0 <= i1 & i1 < sa..(old csize))" */
7093
7094        int r2b = sb.indexOf(v2);
7095        sb.set(i1, v1);
```

```
7096
7097        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
7098    }
7099
7100    static void set_indexOf_post_c_197(ArrayList sa, ArrayList sb, int i1, Object v1,
            Object v2)
7101    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7102                  sa..contents = sb..contents & sa..csize = sb..csize"
7103        modifies "sa..contents", "sb..contents"
7104        ensures "True" */
7105    {
7106        /*: assume "0 <= i1 & i1 < sa..csize" */
7107        sa.set(i1, v1);
7108        int r2a = sa.indexOf(v2);
7109        /*: assume "~((r2a < 0 & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 <
                sa..(old csize)) | (0 <= r2a & r2a < i1) | (r2a = i1 & (i1, v2) : sa..(old
                contents) & 0 <= i1 & i1 < sa..(old csize)) | (r2a > i1 & (i1, v2) ~:
                sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)))" */
7110
7111        int r2b = sb.indexOf(v2);
7112        /*: assume "0 <= i1 & i1 < sb..csize" */
7113        sb.set(i1, v1);
7114
7115        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
7116    }
7117
7118    static void set_lastIndexOf_pre_s_198(ArrayList sa, ArrayList sb, int i1, Object
            v1, Object v2)
7119    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7120                  sa..contents = sb..contents & sa..csize = sb..csize"
7121        modifies "sa..contents", "sb..contents"
7122        ensures "True" */
7123    {
7124        /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
                v2) | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
                sa..contents & i1 <= i & i < sa..csize) & v1 ~= v2) | ((i1, v2) :
                sa..contents & 0 <= i1 & i1 < sa..csize & ~(EX i. (i, v2) : sa..contents &
                i1 < i & i < sa..csize) & v1 = v2) | (EX i. (i, v2) : sa..contents & i1 < i
                & i < sa..csize)" */
7125        /*: assume "0 <= i1 & i1 < sa..csize" */
7126        sa.set(i1, v1);
7127        int r2a = sa.lastIndexOf(v2);
7128
7129        int r2b = sb.lastIndexOf(v2);
7130        sb.set(i1, v1);
7131
7132        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
7133    }
7134
7135    static void set_lastIndexOf_pre_c_198(ArrayList sa, ArrayList sb, int i1, Object
            v1, Object v2)
7136    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7137                  sa..contents = sb..contents & sa..csize = sb..csize"
7138        modifies "sa..contents", "sb..contents"
7139        ensures "True" */
7140    {
7141        /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
                v2) | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
                sa..contents & i1 <= i & i < sa..csize) & v1 ~= v2) | ((i1, v2) :
                sa..contents & 0 <= i1 & i1 < sa..csize & ~(EX i. (i, v2) : sa..contents &
                i1 < i & i < sa..csize) & v1 = v2) | (EX i. (i, v2) : sa..contents & i1 < i
                & i < sa..csize))" */
7142        /*: assume "0 <= i1 & i1 < sa..csize" */
7143        sa.set(i1, v1);
```

```
7144        int r2a = sa.lastIndexOf(v2);
7145
7146        int r2b = sb.lastIndexOf(v2);
7147        /*: assume "0 <= i1 & i1 < sb..csize" */
7148        sb.set(i1, v1);
7149
7150        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
7151    }
7152
7153    static void set_lastIndexOf_between_s_199(ArrayList sa, ArrayList sb, int i1,
            Object v1, Object v2)
7154    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7155                  sa..contents = sb..contents & sa..csize = sb..csize"
7156        modifies "sa..contents", "sb..contents"
7157        ensures "True" */
7158    {
7159        /*: assume "0 <= i1 & i1 < sa..csize" */
7160        sa.set(i1, v1);
7161        /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & (i1,
                v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | ((EX i. (i,
                v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) : sa..contents & i1
                <= i & i < sa..csize) & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 <
                sa..(old csize)) | ((i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize &
                ~(EX i. (i, v2) : sa..contents & i1 < i & i < sa..csize) & (i1, v2) :
                sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | (EX i. (i, v2) :
                sa..contents & i1 < i & i < sa..csize)" */
7162        int r2a = sa.lastIndexOf(v2);
7163
7164        int r2b = sb.lastIndexOf(v2);
7165        sb.set(i1, v1);
7166
7167        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
7168    }
7169
7170    static void set_lastIndexOf_between_c_199(ArrayList sa, ArrayList sb, int i1,
            Object v1, Object v2)
7171    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7172                  sa..contents = sb..contents & sa..csize = sb..csize"
7173        modifies "sa..contents", "sb..contents"
7174        ensures "True" */
7175    {
7176        /*: assume "0 <= i1 & i1 < sa..csize" */
7177        sa.set(i1, v1);
7178        /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & (i1,
                v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | ((EX i. (i,
                v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) : sa..contents & i1
                <= i & i < sa..csize) & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 <
                sa..(old csize)) | ((i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize &
                ~(EX i. (i, v2) : sa..contents & i1 < i & i < sa..csize) & (i1, v2) :
                sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | (EX i. (i, v2) :
                sa..contents & i1 < i & i < sa..csize))" */
7179        int r2a = sa.lastIndexOf(v2);
7180
7181        int r2b = sb.lastIndexOf(v2);
7182        /*: assume "0 <= i1 & i1 < sb..csize" */
7183        sb.set(i1, v1);
7184
7185        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
7186    }
7187
7188    static void set_lastIndexOf_post_s_200(ArrayList sa, ArrayList sb, int i1, Object
            v1, Object v2)
7189    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
```

```
7190                 sa..contents = sb..contents & sa..csize = sb..csize"
7191         modifies "sa..contents", "sb..contents"
7192         ensures "True" */
7193     {
7194         /*: assume "0 <= i1 & i1 < sa..csize" */
7195         sa.set(i1, v1);
7196         int r2a = sa.lastIndexOf(v2);
7197         /*: assume "(r2a < 0 & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old
                 csize)) | (0 <= r2a & r2a < i1 & (i1, v2) ~: sa..(old contents) & 0 <= i1 &
                 i1 < sa..(old csize)) | (r2a = i1 & (i1, v2) : sa..(old contents) & 0 <= i1
                 & i1 < sa..(old csize)) | r2a > i1" */
7198
7199         int r2b = sb.lastIndexOf(v2);
7200         sb.set(i1, v1);
7201
7202         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
7203     }
7204
7205     static void set_lastIndexOf_post_c_200(ArrayList sa, ArrayList sb, int i1, Object
            v1, Object v2)
7206     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7207                 sa..contents = sb..contents & sa..csize = sb..csize"
7208         modifies "sa..contents", "sb..contents"
7209         ensures "True" */
7210     {
7211         /*: assume "0 <= i1 & i1 < sa..csize" */
7212         sa.set(i1, v1);
7213         int r2a = sa.lastIndexOf(v2);
7214         /*: assume "~((r2a < 0 & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 <
                 sa..(old csize)) | (0 <= r2a & r2a < i1 & (i1, v2) ~: sa..(old contents) & 0
                 <= i1 & i1 < sa..(old csize)) | (r2a = i1 & (i1, v2) : sa..(old contents) &
                 0 <= i1 & i1 < sa..(old csize)) | r2a > i1)" */
7215
7216         int r2b = sb.lastIndexOf(v2);
7217         /*: assume "0 <= i1 & i1 < sb..csize" */
7218         sb.set(i1, v1);
7219
7220         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                 */
7221     }
7222
7223     static void set_remove_at_pre_s_201(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2)
7224     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7225                 sa..contents = sb..contents & sa..csize = sb..csize"
7226         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7227         ensures "True" */
7228     {
7229         /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (ALL v. ((i1, v) :
                 sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
                 (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
                 1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
                 sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
                 (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
                 1 < sa..csize)" */
7230         /*: assume "0 <= i1 & i1 < sa..csize" */
7231         sa.set(i1, v1);
7232         /*: assume "0 <= i2 & i2 < sa..csize" */
7233         Object r2a = sa.remove_at(i2);
7234
7235         Object r2b = sb.remove_at(i2);
7236         sb.set(i1, v1);
7237
7238         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
7239     }
```

```
7240
7241    static void set_remove_at_pre_c_201(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2)
7242    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7243                sa..contents = sb..contents & sa..csize = sb..csize"
7244        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7245        ensures "True" */
7246    {
7247        /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (ALL v. ((i1, v) :
                sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
                (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
                1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
                sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
                (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
                1 < sa..csize))" */
7248        /*: assume "0 <= i1 & i1 < sa..csize" */
7249        sa.set(i1, v1);
7250        /*: assume "0 <= i2 & i2 < sa..csize" */
7251        Object r2a = sa.remove_at(i2);
7252
7253        /*: assume "0 <= i2 & i2 < sb..csize" */
7254        Object r2b = sb.remove_at(i2);
7255        /*: assume "0 <= i1 & i1 < sb..csize" */
7256        sb.set(i1, v1);
7257
7258        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
7259    }
7260
7261    static void set_remove_at_between_s_202(ArrayList sa, ArrayList sb, int i1, Object
            v1, int i2)
7262    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7263                sa..contents = sb..contents & sa..csize = sb..csize"
7264        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7265        ensures "True" */
7266    {
7267        /*: assume "0 <= i1 & i1 < sa..csize" */
7268        sa.set(i1, v1);
7269        /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (ALL v. ((i1, v) :
                sa..(old contents)) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..(old
                contents) & (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) & 0
                <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v.
                ((i1, v) : sa..(old contents)) = ((i1 + 1, v) : sa..contents)) & (i1, v1) :
                sa..(old contents) & (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old
                csize) & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
7270        /*: assume "0 <= i2 & i2 < sa..csize" */
7271        Object r2a = sa.remove_at(i2);
7272
7273        Object r2b = sb.remove_at(i2);
7274        sb.set(i1, v1);
7275
7276        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
7277    }
7278
7279    static void set_remove_at_between_c_202(ArrayList sa, ArrayList sb, int i1, Object
            v1, int i2)
7280    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7281                sa..contents = sb..contents & sa..csize = sb..csize"
7282        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7283        ensures "True" */
7284    {
7285        /*: assume "0 <= i1 & i1 < sa..csize" */
7286        sa.set(i1, v1);
7287        /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (ALL v. ((i1, v) :
                sa..(old contents)) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..(old
```

```
            contents) & (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) & 0
            <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v.
            ((i1, v) : sa..(old contents)) = ((i1 + 1, v) : sa..contents)) & (i1, v1) :
            sa..(old contents) & (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old
            csize) & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
7288        /*: assume "0 <= i2 & i2 < sa..csize" */
7289        Object r2a = sa.remove_at(i2);
7290
7291        /*: assume "0 <= i2 & i2 < sb..csize" */
7292        Object r2b = sb.remove_at(i2);
7293        /*: assume "0 <= i1 & i1 < sb..csize" */
7294        sb.set(i1, v1);
7295
7296        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
7297    }
7298
7299    static void set_remove_at_post_s_203(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2)
7300    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7301                  sa..contents = sb..contents & sa..csize = sb..csize"
7302        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7303        ensures "True" */
7304    {
7305        /*: assume "0 <= i1 & i1 < sa..csize" */
7306        sa.set(i1, v1);
7307        /*: assume "0 <= i2 & i2 < sa..csize" */
7308        Object r2a = sa.remove_at(i2);
7309        /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize & (ALL v. ((i1, v) : sa..(old
            contents)) = ((i1, v) : sa..contents)) & (i1, v1) : sa..(old contents) &
            (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 & i1 <
            sa..csize) | (sa..csize > i1 & i1 > i2 & (ALL v. ((i1, v) : sa..(old
            contents)) = ((i1, v) : sa..contents)) & (i1, v1) : sa..(old contents) &
            (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 & i1 <
            sa..csize)" */
7310
7311        Object r2b = sb.remove_at(i2);
7312        sb.set(i1, v1);
7313
7314        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
7315    }
7316
7317    static void set_remove_at_post_c_203(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2)
7318    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7319                  sa..contents = sb..contents & sa..csize = sb..csize"
7320        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7321        ensures "True" */
7322    {
7323        /*: assume "0 <= i1 & i1 < sa..csize" */
7324        sa.set(i1, v1);
7325        /*: assume "0 <= i2 & i2 < sa..csize" */
7326        Object r2a = sa.remove_at(i2);
7327        /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize & (ALL v. ((i1, v) : sa..(old
            contents)) = ((i1, v) : sa..contents)) & (i1, v1) : sa..(old contents) &
            (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 & i1 <
            sa..csize) | (sa..csize > i1 & i1 > i2 & (ALL v. ((i1, v) : sa..(old
            contents)) = ((i1, v) : sa..contents)) & (i1, v1) : sa..(old contents) &
            (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 & i1 <
            sa..csize))" */
7328
7329        /*: assume "0 <= i2 & i2 < sb..csize" */
7330        Object r2b = sb.remove_at(i2);
7331        /*: assume "0 <= i1 & i1 < sb..csize" */
7332        sb.set(i1, v1);
```

```
7333
7334        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
7335    }
7336
7337    static void set_remove_at_pre_s_204(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2)
7338    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7339                  sa..contents = sb..contents & sa..csize = sb..csize"
7340        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7341        ensures "True" */
7342    {
7343        /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, v1) :
                sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
                > i2 & (ALL v. ((i1, v) : sa..contents) = ((i1 + 1, v) : sa..contents)) &
                (i1, v1) : sa..contents & (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 <
                sa..csize & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
7344        /*: assume "0 <= i1 & i1 < sa..csize" */
7345        sa.set(i1, v1);
7346        /*: assume "0 <= i2 & i2 < sa..csize" */
7347        sa.remove_at(i2);
7348
7349        sb.remove_at(i2);
7350        sb.set(i1, v1);
7351
7352        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
7353    }
7354
7355    static void set_remove_at_pre_c_204(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2)
7356    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7357                  sa..contents = sb..contents & sa..csize = sb..csize"
7358        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7359        ensures "True" */
7360    {
7361        /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, v1) :
                sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
                > i2 & (ALL v. ((i1, v) : sa..contents) = ((i1 + 1, v) : sa..contents)) &
                (i1, v1) : sa..contents & (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 <
                sa..csize & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
7362        /*: assume "0 <= i1 & i1 < sa..csize" */
7363        sa.set(i1, v1);
7364        /*: assume "0 <= i2 & i2 < sa..csize" */
7365        sa.remove_at(i2);
7366
7367        /*: assume "0 <= i2 & i2 < sb..csize" */
7368        sb.remove_at(i2);
7369        /*: assume "0 <= i1 & i1 < sb..csize" */
7370        sb.set(i1, v1);
7371
7372        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
7373    }
7374
7375    static void set_remove_at_between_s_205(ArrayList sa, ArrayList sb, int i1, Object
            v1, int i2)
7376    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7377                  sa..contents = sb..contents & sa..csize = sb..csize"
7378        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7379        ensures "True" */
7380    {
7381        /*: assume "0 <= i1 & i1 < sa..csize" */
7382        sa.set(i1, v1);
7383        /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, v1) :
                sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
                > i2 & (ALL v. ((i1, v) : sa..(old contents)) = ((i1 + 1, v) :
```

```
        sa..contents)) & (i1, v1) : sa..(old contents) & (i1 + 1, v1) : sa..contents
        & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
    /*: assume "0 <= i2 & i2 < sa..csize" */
    sa.remove_at(i2);

    sb.remove_at(i2);
    sb.set(i1, v1);

    /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
}

static void set_remove_at_between_c_205(ArrayList sa, ArrayList sb, int i1, Object
    v1, int i2)
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
              sa..contents = sb..contents & sa..csize = sb..csize"
    modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
    ensures "True" */
{
    /*: assume "0 <= i1 & i1 < sa..csize" */
    sa.set(i1, v1);
    /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, v1) :
        sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
        > i2 & (ALL v. ((i1, v) : sa..(old contents)) = ((i1 + 1, v) :
        sa..contents)) & (i1, v1) : sa..(old contents) & (i1 + 1, v1) : sa..contents
        & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
    /*: assume "0 <= i2 & i2 < sa..csize" */
    sa.remove_at(i2);

    /*: assume "0 <= i2 & i2 < sb..csize" */
    sb.remove_at(i2);
    /*: assume "0 <= i1 & i1 < sb..csize" */
    sb.set(i1, v1);

    /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
}

static void set_remove_at_post_s_206(ArrayList sa, ArrayList sb, int i1, Object v1,
    int i2)
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
              sa..contents = sb..contents & sa..csize = sb..csize"
    modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
    ensures "True" */
{
    /*: assume "0 <= i1 & i1 < sa..csize" */
    sa.set(i1, v1);
    /*: assume "0 <= i2 & i2 < sa..csize" */
    sa.remove_at(i2);
    /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize & (i1, v1) : sa..contents & 0
        <= i1 & i1 < sa..csize) | (sa..csize > i1 & i1 > i2 & (ALL v. ((i1, v) :
        sa..(old contents)) = ((i1, v) : sa..(contents))) & (i1, v1) : sa..(old
        contents) & (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) & 0 <=
        i1 & i1 < sa..csize)" */

    sb.remove_at(i2);
    sb.set(i1, v1);

    /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
}

static void set_remove_at_post_c_206(ArrayList sa, ArrayList sb, int i1, Object v1,
    int i2)
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
              sa..contents = sb..contents & sa..csize = sb..csize"
    modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
    ensures "True" */
```

```
7436    {
7437        /*: assume "0 <= i1 & i1 < sa..csize" */
7438        sa.set(i1, v1);
7439        /*: assume "0 <= i2 & i2 < sa..csize" */
7440        sa.remove_at(i2);
7441        /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize & (i1, v1) : sa..contents & 0
                <= i1 & i1 < sa..csize) | (sa..csize > i1 & i1 > i2 & (ALL v. ((i1, v) :
                sa..(old contents)) = ((i1, v) : sa..(contents))) & (i1, v1) : sa..(old
                contents) & (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) & 0 <=
                i1 & i1 < sa..csize))" */
7442
7443        /*: assume "0 <= i2 & i2 < sb..csize" */
7444        sb.remove_at(i2);
7445        /*: assume "0 <= i1 & i1 < sb..csize" */
7446        sb.set(i1, v1);
7447
7448        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
7449    }
7450
7451    static void set_set_pre_s_207(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2, Object v2)
7452    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7453                  sa..contents = sb..contents & sa..csize = sb..csize"
7454        modifies "sa..contents", "sb..contents"
7455        ensures "True" */
7456    {
7457        /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
                sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | i1 > i2" */
7458        /*: assume "0 <= i1 & i1 < sa..csize" */
7459        sa.set(i1, v1);
7460        /*: assume "0 <= i2 & i2 < sa..csize" */
7461        Object r2a = sa.set(i2, v2);
7462
7463        Object r2b = sb.set(i2, v2);
7464        sb.set(i1, v1);
7465
7466        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
7467    }
7468
7469    static void set_set_pre_c_207(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2, Object v2)
7470    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7471                  sa..contents = sb..contents & sa..csize = sb..csize"
7472        modifies "sa..contents", "sb..contents"
7473        ensures "True" */
7474    {
7475        /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
                sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | i1 > i2)" */
7476        /*: assume "0 <= i1 & i1 < sa..csize" */
7477        sa.set(i1, v1);
7478        /*: assume "0 <= i2 & i2 < sa..csize" */
7479        Object r2a = sa.set(i2, v2);
7480
7481        /*: assume "0 <= i2 & i2 < sb..csize" */
7482        Object r2b = sb.set(i2, v2);
7483        /*: assume "0 <= i1 & i1 < sb..csize" */
7484        sb.set(i1, v1);
7485
7486        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
7487    }
7488
7489    static void set_set_between_s_208(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2, Object v2)
7490    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
```

```
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.set(i1, v1);
        /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & (i1, v2) :
            sa..(old contents) & v1 = v2 & 0 <= i1 & i1 < sa..(old csize)) | i1 > i2" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        Object r2b = sb.set(i2, v2);
        sb.set(i1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void set_set_between_c_208(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.set(i1, v1);
        /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & (i1, v2) :
            sa..(old contents) & v1 = v2 & 0 <= i1 & i1 < sa..(old csize)) | i1 > i2)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.set(i1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void set_set_post_s_209(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);
        /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & (i1, v2) :
            sa..(old contents) & v1 = v2 & 0 <= i1 & i1 < sa..(old csize)) | i1 > i2" */

        Object r2b = sb.set(i2, v2);
        sb.set(i1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void set_set_post_c_209(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
```

```
7549        ensures "True" */
7550    {
7551        /*: assume "0 <= i1 & i1 < sa..csize" */
7552        sa.set(i1, v1);
7553        /*: assume "0 <= i2 & i2 < sa..csize" */
7554        Object r2a = sa.set(i2, v2);
7555        /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & (i1, v2) :
                sa..(old contents) & v1 = v2 & 0 <= i1 & i1 < sa..(old csize)) | i1 > i2)" */
7556
7557        /*: assume "0 <= i2 & i2 < sb..csize" */
7558        Object r2b = sb.set(i2, v2);
7559        /*: assume "0 <= i1 & i1 < sb..csize" */
7560        sb.set(i1, v1);
7561
7562        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
7563    }
7564
7565    static void set_set_pre_s_210(ArrayList sa, ArrayList sb, int i1, Object v1, int
           i2, Object v2)
7566    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7567                  sa..contents = sb..contents & sa..csize = sb..csize"
7568        modifies "sa..contents", "sb..contents"
7569        ensures "True" */
7570    {
7571        /*: assume "i1 < i2 | (i1 = i2 & v1 = v2) | i1 > i2" */
7572        /*: assume "0 <= i1 & i1 < sa..csize" */
7573        sa.set(i1, v1);
7574        /*: assume "0 <= i2 & i2 < sa..csize" */
7575        sa.set(i2, v2);
7576
7577        sb.set(i2, v2);
7578        sb.set(i1, v1);
7579
7580        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
7581    }
7582
7583    static void set_set_pre_c_210(ArrayList sa, ArrayList sb, int i1, Object v1, int
           i2, Object v2)
7584    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7585                  sa..contents = sb..contents & sa..csize = sb..csize"
7586        modifies "sa..contents", "sb..contents"
7587        ensures "True" */
7588    {
7589        /*: assume "~(i1 < i2 | (i1 = i2 & v1 = v2) | i1 > i2)" */
7590        /*: assume "0 <= i1 & i1 < sa..csize" */
7591        sa.set(i1, v1);
7592        /*: assume "0 <= i2 & i2 < sa..csize" */
7593        sa.set(i2, v2);
7594
7595        /*: assume "0 <= i2 & i2 < sb..csize" */
7596        sb.set(i2, v2);
7597        /*: assume "0 <= i1 & i1 < sb..csize" */
7598        sb.set(i1, v1);
7599
7600        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
7601    }
7602
7603    static void set_set_between_s_211(ArrayList sa, ArrayList sb, int i1, Object v1,
           int i2, Object v2)
7604    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7605                  sa..contents = sb..contents & sa..csize = sb..csize"
7606        modifies "sa..contents", "sb..contents"
7607        ensures "True" */
7608    {
```

```
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.set(i1, v1);
        /*: assume "i1 < i2 | (i1 = i2 & v1 = v2) | i1 > i2" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        sb.set(i2, v2);
        sb.set(i1, v1);

        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void set_set_between_c_211(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.set(i1, v1);
        /*: assume "~(i1 < i2 | (i1 = i2 & v1 = v2) | i1 > i2)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.set(i1, v1);

        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
    }

    static void set_set_post_s_212(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);
        /*: assume "i1 < i2 | (i1 = i2 & v1 = v2) | i1 > i2" */

        sb.set(i2, v2);
        sb.set(i1, v1);

        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void set_set_post_c_212(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);
        /*: assume "~(i1 < i2 | (i1 = i2 & v1 = v2) | i1 > i2)" */

```

```
        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.set(i1, v1);

        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
    }

    static void set_size_pre_s_213(ArrayList sa, ArrayList sb, int i1, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "True" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.set(i1, v1);
        int r2a = sa.size();

        int r2b = sb.size();
        sb.set(i1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void set_size_pre_c_213(ArrayList sa, ArrayList sb, int i1, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "~(True)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.set(i1, v1);
        int r2a = sa.size();

        int r2b = sb.size();
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.set(i1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void set_size_between_s_214(ArrayList sa, ArrayList sb, int i1, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.set(i1, v1);
        /*: assume "True" */
        int r2a = sa.size();

        int r2b = sb.size();
        sb.set(i1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void set_size_between_c_214(ArrayList sa, ArrayList sb, int i1, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
```

```
7735        ensures "True" */
7736    {
7737        /*: assume "0 <= i1 & i1 < sa..csize" */
7738        sa.set(i1, v1);
7739        /*: assume "~(True)" */
7740        int r2a = sa.size();
7741
7742        int r2b = sb.size();
7743        /*: assume "0 <= i1 & i1 < sb..csize" */
7744        sb.set(i1, v1);
7745
7746        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
7747    }
7748
7749    static void set_size_post_s_215(ArrayList sa, ArrayList sb, int i1, Object v1)
7750    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7751                  sa..contents = sb..contents & sa..csize = sb..csize"
7752        modifies "sa..contents", "sb..contents"
7753        ensures "True" */
7754    {
7755        /*: assume "0 <= i1 & i1 < sa..csize" */
7756        sa.set(i1, v1);
7757        int r2a = sa.size();
7758        /*: assume "True" */
7759
7760        int r2b = sb.size();
7761        sb.set(i1, v1);
7762
7763        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
7764    }
7765
7766    static void set_size_post_c_215(ArrayList sa, ArrayList sb, int i1, Object v1)
7767    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7768                  sa..contents = sb..contents & sa..csize = sb..csize"
7769        modifies "sa..contents", "sb..contents"
7770        ensures "True" */
7771    {
7772        /*: assume "0 <= i1 & i1 < sa..csize" */
7773        sa.set(i1, v1);
7774        int r2a = sa.size();
7775        /*: assume "~(True)" */
7776
7777        int r2b = sb.size();
7778        /*: assume "0 <= i1 & i1 < sb..csize" */
7779        sb.set(i1, v1);
7780
7781        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
7782    }
7783
7784    static void size_add_at_pre_s_216(ArrayList sa, ArrayList sb, int i2, Object v2)
7785    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7786                  sa..contents = sb..contents & sa..csize = sb..csize"
7787        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
7788        ensures "True" */
7789    {
7790        /*: assume "False" */
7791        int r1a = sa.size();
7792        /*: assume "0 <= i2 & i2 <= sa..csize" */
7793        sa.add_at(i2, v2);
7794
7795        sb.add_at(i2, v2);
7796        int r1b = sb.size();
```

```
7797
7798        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
7799    }
7800
7801    static void size_add_at_pre_c_216(ArrayList sa, ArrayList sb, int i2, Object v2)
7802    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7803                  sa..contents = sb..contents & sa..csize = sb..csize"
7804        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
7805        ensures "True" */
7806    {
7807        /*: assume "~(False)" */
7808        int r1a = sa.size();
7809        /*: assume "0 <= i2 & i2 <= sa..csize" */
7810        sa.add_at(i2, v2);
7811
7812        /*: assume "0 <= i2 & i2 <= sb..csize" */
7813        sb.add_at(i2, v2);
7814        int r1b = sb.size();
7815
7816        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
7817    }
7818
7819    static void size_add_at_between_s_217(ArrayList sa, ArrayList sb, int i2, Object v2)
7820    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7821                  sa..contents = sb..contents & sa..csize = sb..csize"
7822        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
7823        ensures "True" */
7824    {
7825        int r1a = sa.size();
7826        /*: assume "False" */
7827        /*: assume "0 <= i2 & i2 <= sa..csize" */
7828        sa.add_at(i2, v2);
7829
7830        sb.add_at(i2, v2);
7831        int r1b = sb.size();
7832
7833        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
7834    }
7835
7836    static void size_add_at_between_c_217(ArrayList sa, ArrayList sb, int i2, Object v2)
7837    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7838                  sa..contents = sb..contents & sa..csize = sb..csize"
7839        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
7840        ensures "True" */
7841    {
7842        int r1a = sa.size();
7843        /*: assume "~(False)" */
7844        /*: assume "0 <= i2 & i2 <= sa..csize" */
7845        sa.add_at(i2, v2);
7846
7847        /*: assume "0 <= i2 & i2 <= sb..csize" */
7848        sb.add_at(i2, v2);
7849        int r1b = sb.size();
7850
7851        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
7852    }
7853
7854    static void size_add_at_post_s_218(ArrayList sa, ArrayList sb, int i2, Object v2)
7855    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7856                  sa..contents = sb..contents & sa..csize = sb..csize"
```

```
7857        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
7858        ensures "True" */
7859    {
7860        int r1a = sa.size();
7861        /*: assume "0 <= i2 & i2 <= sa..csize" */
7862        sa.add_at(i2, v2);
7863        /*: assume "False" */
7864
7865        sb.add_at(i2, v2);
7866        int r1b = sb.size();
7867
7868        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
7869    }
7870
7871    static void size_add_at_post_c_218(ArrayList sa, ArrayList sb, int i2, Object v2)
7872    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7873                  sa..contents = sb..contents & sa..csize = sb..csize"
7874        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
7875        ensures "True" */
7876    {
7877        int r1a = sa.size();
7878        /*: assume "0 <= i2 & i2 <= sa..csize" */
7879        sa.add_at(i2, v2);
7880        /*: assume "~(False)" */
7881
7882        /*: assume "0 <= i2 & i2 <= sb..csize" */
7883        sb.add_at(i2, v2);
7884        int r1b = sb.size();
7885
7886        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
7887    }
7888
7889    static void size_get_pre_s_219(ArrayList sa, ArrayList sb, int i2)
7890    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7891                  sa..contents = sb..contents & sa..csize = sb..csize"
7892        ensures "True" */
7893    {
7894        /*: assume "True" */
7895        int r1a = sa.size();
7896        /*: assume "0 <= i2 & i2 < sa..csize" */
7897        Object r2a = sa.get(i2);
7898
7899        Object r2b = sb.get(i2);
7900        int r1b = sb.size();
7901
7902        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
7903    }
7904
7905    static void size_get_pre_c_219(ArrayList sa, ArrayList sb, int i2)
7906    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7907                  sa..contents = sb..contents & sa..csize = sb..csize"
7908        ensures "True" */
7909    {
7910        /*: assume "~(True)" */
7911        int r1a = sa.size();
7912        /*: assume "0 <= i2 & i2 < sa..csize" */
7913        Object r2a = sa.get(i2);
7914
7915        /*: assume "0 <= i2 & i2 < sb..csize" */
7916        Object r2b = sb.get(i2);
7917        int r1b = sb.size();
```

```
7918
7919        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
7920    }
7921
7922    static void size_get_between_s_220(ArrayList sa, ArrayList sb, int i2)
7923    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7924                  sa..contents = sb..contents & sa..csize = sb..csize"
7925        ensures "True" */
7926    {
7927        int r1a = sa.size();
7928        /*: assume "True" */
7929        /*: assume "0 <= i2 & i2 < sa..csize" */
7930        Object r2a = sa.get(i2);
7931
7932        Object r2b = sb.get(i2);
7933        int r1b = sb.size();
7934
7935        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
7936    }
7937
7938    static void size_get_between_c_220(ArrayList sa, ArrayList sb, int i2)
7939    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7940                  sa..contents = sb..contents & sa..csize = sb..csize"
7941        ensures "True" */
7942    {
7943        int r1a = sa.size();
7944        /*: assume "~(True)" */
7945        /*: assume "0 <= i2 & i2 < sa..csize" */
7946        Object r2a = sa.get(i2);
7947
7948        /*: assume "0 <= i2 & i2 < sb..csize" */
7949        Object r2b = sb.get(i2);
7950        int r1b = sb.size();
7951
7952        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
7953    }
7954
7955    static void size_get_post_s_221(ArrayList sa, ArrayList sb, int i2)
7956    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7957                  sa..contents = sb..contents & sa..csize = sb..csize"
7958        ensures "True" */
7959    {
7960        int r1a = sa.size();
7961        /*: assume "0 <= i2 & i2 < sa..csize" */
7962        Object r2a = sa.get(i2);
7963        /*: assume "True" */
7964
7965        Object r2b = sb.get(i2);
7966        int r1b = sb.size();
7967
7968        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
7969    }
7970
7971    static void size_get_post_c_221(ArrayList sa, ArrayList sb, int i2)
7972    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7973                  sa..contents = sb..contents & sa..csize = sb..csize"
7974        ensures "True" */
7975    {
7976        int r1a = sa.size();
7977        /*: assume "0 <= i2 & i2 < sa..csize" */
7978        Object r2a = sa.get(i2);
```

```
7979            /*: assume "~(True)" */
7980
7981            /*: assume "0 <= i2 & i2 < sb..csize" */
7982            Object r2b = sb.get(i2);
7983            int r1b = sb.size();
7984
7985            /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                    sb..csize)" */
7986        }
7987
7988        static void size_indexOf_pre_s_222(ArrayList sa, ArrayList sb, Object v2)
7989        /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7990                        sa..contents = sb..contents & sa..csize = sb..csize"
7991            ensures "True" */
7992        {
7993            /*: assume "True" */
7994            int r1a = sa.size();
7995            int r2a = sa.indexOf(v2);
7996
7997            int r2b = sb.indexOf(v2);
7998            int r1b = sb.size();
7999
8000            /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                    sb..csize" */
8001        }
8002
8003        static void size_indexOf_pre_c_222(ArrayList sa, ArrayList sb, Object v2)
8004        /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8005                        sa..contents = sb..contents & sa..csize = sb..csize"
8006            ensures "True" */
8007        {
8008            /*: assume "~(True)" */
8009            int r1a = sa.size();
8010            int r2a = sa.indexOf(v2);
8011
8012            int r2b = sb.indexOf(v2);
8013            int r1b = sb.size();
8014
8015            /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                    sb..csize)" */
8016        }
8017
8018        static void size_indexOf_between_s_223(ArrayList sa, ArrayList sb, Object v2)
8019        /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8020                        sa..contents = sb..contents & sa..csize = sb..csize"
8021            ensures "True" */
8022        {
8023            int r1a = sa.size();
8024            /*: assume "True" */
8025            int r2a = sa.indexOf(v2);
8026
8027            int r2b = sb.indexOf(v2);
8028            int r1b = sb.size();
8029
8030            /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                    sb..csize" */
8031        }
8032
8033        static void size_indexOf_between_c_223(ArrayList sa, ArrayList sb, Object v2)
8034        /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8035                        sa..contents = sb..contents & sa..csize = sb..csize"
8036            ensures "True" */
8037        {
8038            int r1a = sa.size();
8039            /*: assume "~(True)" */
```

```
8040        int r2a = sa.indexOf(v2);
8041
8042        int r2b = sb.indexOf(v2);
8043        int r1b = sb.size();
8044
8045        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
8046    }
8047
8048    static void size_indexOf_post_s_224(ArrayList sa, ArrayList sb, Object v2)
8049    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8050                  sa..contents = sb..contents & sa..csize = sb..csize"
8051        ensures "True" */
8052    {
8053        int r1a = sa.size();
8054        int r2a = sa.indexOf(v2);
8055        /*: assume "True" */
8056
8057        int r2b = sb.indexOf(v2);
8058        int r1b = sb.size();
8059
8060        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
8061    }
8062
8063    static void size_indexOf_post_c_224(ArrayList sa, ArrayList sb, Object v2)
8064    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8065                  sa..contents = sb..contents & sa..csize = sb..csize"
8066        ensures "True" */
8067    {
8068        int r1a = sa.size();
8069        int r2a = sa.indexOf(v2);
8070        /*: assume "~(True)" */
8071
8072        int r2b = sb.indexOf(v2);
8073        int r1b = sb.size();
8074
8075        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
8076    }
8077
8078    static void size_lastIndexOf_pre_s_225(ArrayList sa, ArrayList sb, Object v2)
8079    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8080                  sa..contents = sb..contents & sa..csize = sb..csize"
8081        ensures "True" */
8082    {
8083        /*: assume "True" */
8084        int r1a = sa.size();
8085        int r2a = sa.lastIndexOf(v2);
8086
8087        int r2b = sb.lastIndexOf(v2);
8088        int r1b = sb.size();
8089
8090        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
8091    }
8092
8093    static void size_lastIndexOf_pre_c_225(ArrayList sa, ArrayList sb, Object v2)
8094    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8095                  sa..contents = sb..contents & sa..csize = sb..csize"
8096        ensures "True" */
8097    {
8098        /*: assume "~(True)" */
8099        int r1a = sa.size();
8100        int r2a = sa.lastIndexOf(v2);
```

```
8101
8102        int r2b = sb.lastIndexOf(v2);
8103        int r1b = sb.size();
8104
8105        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
8106    }
8107
8108    static void size_lastIndexOf_between_s_226(ArrayList sa, ArrayList sb, Object v2)
8109    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8110                  sa..contents = sb..contents & sa..csize = sb..csize"
8111        ensures "True" */
8112    {
8113        int r1a = sa.size();
8114        /*: assume "True" */
8115        int r2a = sa.lastIndexOf(v2);
8116
8117        int r2b = sb.lastIndexOf(v2);
8118        int r1b = sb.size();
8119
8120        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
8121    }
8122
8123    static void size_lastIndexOf_between_c_226(ArrayList sa, ArrayList sb, Object v2)
8124    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8125                  sa..contents = sb..contents & sa..csize = sb..csize"
8126        ensures "True" */
8127    {
8128        int r1a = sa.size();
8129        /*: assume "~(True)" */
8130        int r2a = sa.lastIndexOf(v2);
8131
8132        int r2b = sb.lastIndexOf(v2);
8133        int r1b = sb.size();
8134
8135        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
8136    }
8137
8138    static void size_lastIndexOf_post_s_227(ArrayList sa, ArrayList sb, Object v2)
8139    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8140                  sa..contents = sb..contents & sa..csize = sb..csize"
8141        ensures "True" */
8142    {
8143        int r1a = sa.size();
8144        int r2a = sa.lastIndexOf(v2);
8145        /*: assume "True" */
8146
8147        int r2b = sb.lastIndexOf(v2);
8148        int r1b = sb.size();
8149
8150        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
8151    }
8152
8153    static void size_lastIndexOf_post_c_227(ArrayList sa, ArrayList sb, Object v2)
8154    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8155                  sa..contents = sb..contents & sa..csize = sb..csize"
8156        ensures "True" */
8157    {
8158        int r1a = sa.size();
8159        int r2a = sa.lastIndexOf(v2);
8160        /*: assume "~(True)" */
8161
```

```
        int r2b = sb.lastIndexOf(v2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void size_remove_at_pre_s_228(ArrayList sa, ArrayList sb, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "False" */
        int r1a = sa.size();
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);

        Object r2b = sb.remove_at(i2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void size_remove_at_pre_c_228(ArrayList sa, ArrayList sb, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~(False)" */
        int r1a = sa.size();
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.remove_at(i2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void size_remove_at_between_s_229(ArrayList sa, ArrayList sb, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "False" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);

        Object r2b = sb.remove_at(i2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void size_remove_at_between_c_229(ArrayList sa, ArrayList sb, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
```

```
8223        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
8224        ensures "True" */
8225    {
8226        int r1a = sa.size();
8227        /*: assume "~(False)" */
8228        /*: assume "0 <= i2 & i2 < sa..csize" */
8229        Object r2a = sa.remove_at(i2);
8230
8231        /*: assume "0 <= i2 & i2 < sb..csize" */
8232        Object r2b = sb.remove_at(i2);
8233        int r1b = sb.size();
8234
8235        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
8236    }
8237
8238    static void size_remove_at_post_s_230(ArrayList sa, ArrayList sb, int i2)
8239    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8240                  sa..contents = sb..contents & sa..csize = sb..csize"
8241        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
8242        ensures "True" */
8243    {
8244        int r1a = sa.size();
8245        /*: assume "0 <= i2 & i2 < sa..csize" */
8246        Object r2a = sa.remove_at(i2);
8247        /*: assume "False" */
8248
8249        Object r2b = sb.remove_at(i2);
8250        int r1b = sb.size();
8251
8252        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
8253    }
8254
8255    static void size_remove_at_post_c_230(ArrayList sa, ArrayList sb, int i2)
8256    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8257                  sa..contents = sb..contents & sa..csize = sb..csize"
8258        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
8259        ensures "True" */
8260    {
8261        int r1a = sa.size();
8262        /*: assume "0 <= i2 & i2 < sa..csize" */
8263        Object r2a = sa.remove_at(i2);
8264        /*: assume "~(False)" */
8265
8266        /*: assume "0 <= i2 & i2 < sb..csize" */
8267        Object r2b = sb.remove_at(i2);
8268        int r1b = sb.size();
8269
8270        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
8271    }
8272
8273    static void size_remove_at_pre_s_231(ArrayList sa, ArrayList sb, int i2)
8274    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8275                  sa..contents = sb..contents & sa..csize = sb..csize"
8276        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
8277        ensures "True" */
8278    {
8279        /*: assume "False" */
8280        int r1a = sa.size();
8281        /*: assume "0 <= i2 & i2 < sa..csize" */
8282        sa.remove_at(i2);
8283
8284        sb.remove_at(i2);
```

```
        int r1b = sb.size();

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void size_remove_at_pre_c_231(ArrayList sa, ArrayList sb, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~(False)" */
        int r1a = sa.size();
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void size_remove_at_between_s_232(ArrayList sa, ArrayList sb, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "False" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        sb.remove_at(i2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void size_remove_at_between_c_232(ArrayList sa, ArrayList sb, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "~(False)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void size_remove_at_post_s_233(ArrayList sa, ArrayList sb, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
```

```
8348    {
8349        int r1a = sa.size();
8350        /*: assume "0 <= i2 & i2 < sa..csize" */
8351        sa.remove_at(i2);
8352        /*: assume "False" */
8353
8354        sb.remove_at(i2);
8355        int r1b = sb.size();
8356
8357        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
8358    }
8359
8360    static void size_remove_at_post_c_233(ArrayList sa, ArrayList sb, int i2)
8361    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8362                  sa..contents = sb..contents & sa..csize = sb..csize"
8363        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
8364        ensures "True" */
8365    {
8366        int r1a = sa.size();
8367        /*: assume "0 <= i2 & i2 < sa..csize" */
8368        sa.remove_at(i2);
8369        /*: assume "~(False)" */
8370
8371        /*: assume "0 <= i2 & i2 < sb..csize" */
8372        sb.remove_at(i2);
8373        int r1b = sb.size();
8374
8375        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
8376    }
8377
8378    static void size_set_pre_s_234(ArrayList sa, ArrayList sb, int i2, Object v2)
8379    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8380                  sa..contents = sb..contents & sa..csize = sb..csize"
8381        modifies "sa..contents", "sb..contents"
8382        ensures "True" */
8383    {
8384        /*: assume "True" */
8385        int r1a = sa.size();
8386        /*: assume "0 <= i2 & i2 < sa..csize" */
8387        Object r2a = sa.set(i2, v2);
8388
8389        Object r2b = sb.set(i2, v2);
8390        int r1b = sb.size();
8391
8392        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
8393    }
8394
8395    static void size_set_pre_c_234(ArrayList sa, ArrayList sb, int i2, Object v2)
8396    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8397                  sa..contents = sb..contents & sa..csize = sb..csize"
8398        modifies "sa..contents", "sb..contents"
8399        ensures "True" */
8400    {
8401        /*: assume "~(True)" */
8402        int r1a = sa.size();
8403        /*: assume "0 <= i2 & i2 < sa..csize" */
8404        Object r2a = sa.set(i2, v2);
8405
8406        /*: assume "0 <= i2 & i2 < sb..csize" */
8407        Object r2b = sb.set(i2, v2);
8408        int r1b = sb.size();
8409
```

```
        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void size_set_between_s_235(ArrayList sa, ArrayList sb, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "True" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        Object r2b = sb.set(i2, v2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void size_set_between_c_235(ArrayList sa, ArrayList sb, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "~(True)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.set(i2, v2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void size_set_post_s_236(ArrayList sa, ArrayList sb, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);
        /*: assume "True" */

        Object r2b = sb.set(i2, v2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void size_set_post_c_236(ArrayList sa, ArrayList sb, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
```

```
8471            int r1a = sa.size();
8472            /*: assume "0 <= i2 & i2 < sa..csize" */
8473            Object r2a = sa.set(i2, v2);
8474            /*: assume "~(True)" */
8475
8476            /*: assume "0 <= i2 & i2 < sb..csize" */
8477            Object r2b = sb.set(i2, v2);
8478            int r1b = sb.size();
8479
8480            /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                    sb..csize)" */
8481        }
8482
8483        static void size_set_pre_s_237(ArrayList sa, ArrayList sb, int i2, Object v2)
8484        /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8485                      sa..contents = sb..contents & sa..csize = sb..csize"
8486            modifies "sa..contents", "sb..contents"
8487            ensures "True" */
8488        {
8489            /*: assume "True" */
8490            int r1a = sa.size();
8491            /*: assume "0 <= i2 & i2 < sa..csize" */
8492            sa.set(i2, v2);
8493
8494            sb.set(i2, v2);
8495            int r1b = sb.size();
8496
8497            /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
8498        }
8499
8500        static void size_set_pre_c_237(ArrayList sa, ArrayList sb, int i2, Object v2)
8501        /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8502                      sa..contents = sb..contents & sa..csize = sb..csize"
8503            modifies "sa..contents", "sb..contents"
8504            ensures "True" */
8505        {
8506            /*: assume "~(True)" */
8507            int r1a = sa.size();
8508            /*: assume "0 <= i2 & i2 < sa..csize" */
8509            sa.set(i2, v2);
8510
8511            /*: assume "0 <= i2 & i2 < sb..csize" */
8512            sb.set(i2, v2);
8513            int r1b = sb.size();
8514
8515            /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                    */
8516        }
8517
8518        static void size_set_between_s_238(ArrayList sa, ArrayList sb, int i2, Object v2)
8519        /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8520                      sa..contents = sb..contents & sa..csize = sb..csize"
8521            modifies "sa..contents", "sb..contents"
8522            ensures "True" */
8523        {
8524            int r1a = sa.size();
8525            /*: assume "True" */
8526            /*: assume "0 <= i2 & i2 < sa..csize" */
8527            sa.set(i2, v2);
8528
8529            sb.set(i2, v2);
8530            int r1b = sb.size();
8531
8532            /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
8533        }
```

```
8534
8535    static void size_set_between_c_238(ArrayList sa, ArrayList sb, int i2, Object v2)
8536    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8537                sa..contents = sb..contents & sa..csize = sb..csize"
8538        modifies "sa..contents", "sb..contents"
8539        ensures "True" */
8540    {
8541        int r1a = sa.size();
8542        /*: assume "~(True)" */
8543        /*: assume "0 <= i2 & i2 < sa..csize" */
8544        sa.set(i2, v2);
8545
8546        /*: assume "0 <= i2 & i2 < sb..csize" */
8547        sb.set(i2, v2);
8548        int r1b = sb.size();
8549
8550        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
8551    }
8552
8553    static void size_set_post_s_239(ArrayList sa, ArrayList sb, int i2, Object v2)
8554    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8555                sa..contents = sb..contents & sa..csize = sb..csize"
8556        modifies "sa..contents", "sb..contents"
8557        ensures "True" */
8558    {
8559        int r1a = sa.size();
8560        /*: assume "0 <= i2 & i2 < sa..csize" */
8561        sa.set(i2, v2);
8562        /*: assume "True" */
8563
8564        sb.set(i2, v2);
8565        int r1b = sb.size();
8566
8567        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
8568    }
8569
8570    static void size_set_post_c_239(ArrayList sa, ArrayList sb, int i2, Object v2)
8571    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8572                sa..contents = sb..contents & sa..csize = sb..csize"
8573        modifies "sa..contents", "sb..contents"
8574        ensures "True" */
8575    {
8576        int r1a = sa.size();
8577        /*: assume "0 <= i2 & i2 < sa..csize" */
8578        sa.set(i2, v2);
8579        /*: assume "~(True)" */
8580
8581        /*: assume "0 <= i2 & i2 < sb..csize" */
8582        sb.set(i2, v2);
8583        int r1b = sb.size();
8584
8585        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
8586    }
8587
8588    static void size_size_pre_s_240(ArrayList sa, ArrayList sb)
8589    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8590                sa..contents = sb..contents & sa..csize = sb..csize"
8591        ensures "True" */
8592    {
8593        /*: assume "True" */
8594        int r1a = sa.size();
8595        int r2a = sa.size();
8596
```

```
8597        int r2b = sb.size();
8598        int r1b = sb.size();
8599
8600        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
8601    }
8602
8603    static void size_size_pre_c_240(ArrayList sa, ArrayList sb)
8604    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8605                  sa..contents = sb..contents & sa..csize = sb..csize"
8606        ensures "True" */
8607    {
8608        /*: assume "~(True)" */
8609        int r1a = sa.size();
8610        int r2a = sa.size();
8611
8612        int r2b = sb.size();
8613        int r1b = sb.size();
8614
8615        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
8616    }
8617
8618    static void size_size_between_s_241(ArrayList sa, ArrayList sb)
8619    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8620                  sa..contents = sb..contents & sa..csize = sb..csize"
8621        ensures "True" */
8622    {
8623        int r1a = sa.size();
8624        /*: assume "True" */
8625        int r2a = sa.size();
8626
8627        int r2b = sb.size();
8628        int r1b = sb.size();
8629
8630        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
8631    }
8632
8633    static void size_size_between_c_241(ArrayList sa, ArrayList sb)
8634    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8635                  sa..contents = sb..contents & sa..csize = sb..csize"
8636        ensures "True" */
8637    {
8638        int r1a = sa.size();
8639        /*: assume "~(True)" */
8640        int r2a = sa.size();
8641
8642        int r2b = sb.size();
8643        int r1b = sb.size();
8644
8645        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
8646    }
8647
8648    static void size_size_post_s_242(ArrayList sa, ArrayList sb)
8649    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8650                  sa..contents = sb..contents & sa..csize = sb..csize"
8651        ensures "True" */
8652    {
8653        int r1a = sa.size();
8654        int r2a = sa.size();
8655        /*: assume "True" */
8656
8657        int r2b = sb.size();
```

```
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void size_size_post_c_242(ArrayList sa, ArrayList sb)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                   sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.size();
        int r2a = sa.size();
        /*: assume "~(True)" */

        int r2b = sb.size();
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

}
```

Listing 18. ArrayListComm.java

```
class ArrayListComm {
    static void add_at_add_at_pre_s_0(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                   sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "(i1 < i2 & i2 <= sa..csize & (i2 - 1, v2) : sa..contents & 0 <= i2
            - 1 & i2 - 1 < sa..csize) | (i1 = i2 & v1 = v2) | (i1 > i2 & (i1 - 1, v1) :
            sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize)" */
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        sb.add_at(i2, v2);
        sb.add_at(i1, v1);

        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void add_at_add_at_pre_c_0(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                   sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "~((i1 < i2 & i2 <= sa..csize & (i2 - 1, v2) : sa..contents & 0 <=
            i2 - 1 & i2 - 1 < sa..csize) | (i1 = i2 & v1 = v2) | (i1 > i2 & (i1 - 1, v1)
            : sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize))" */
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

```

```
        /*: assume "0 <= i2 & i2 <= sb..csize" */
        sb.add_at(i2, v2);
        /*: assume "0 <= i1 & i1 <= sb..csize" */
        sb.add_at(i1, v1);

        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
    }

    static void add_at_add_at_between_s_1(ArrayList sa, ArrayList sb, int i1, Object
        v1, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "(i1 < i2 & i2 <= sa..csize - 1 & (i2, v2) : sa..contents & 0 <= i2
            & i2 < sa..csize) | (i1 = i2 & v1 = v2) | (i1 > i2 & (i1 - 1, v1) :
            sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize)" */
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        sb.add_at(i2, v2);
        sb.add_at(i1, v1);

        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void add_at_add_at_between_c_1(ArrayList sa, ArrayList sb, int i1, Object
        v1, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "~((i1 < i2 & i2 <= sa..csize - 1 & (i2, v2) : sa..contents & 0 <=
            i2 & i2 < sa..csize) | (i1 = i2 & v1 = v2) | (i1 > i2 & (i1 - 1, v1) :
            sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize))" */
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        /*: assume "0 <= i2 & i2 <= sb..csize" */
        sb.add_at(i2, v2);
        /*: assume "0 <= i1 & i1 <= sb..csize" */
        sb.add_at(i1, v1);

        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
    }

    static void add_at_add_at_post_s_2(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
```

```
87          sa.add_at(i2, v2);
88          /*: assume "(i1 < i2 & i2 <= sa..csize - 2 & (i2 + 1, v2) : sa..contents & 0 <=
                i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & v1 = v2) | (i1 > i2 & (i1, v1) :
                sa..contents & 0 <= i1 & i1 < sa..csize)" */
89
90          sb.add_at(i2, v2);
91          sb.add_at(i1, v1);
92
93          /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
94      }
95
96      static void add_at_add_at_post_c_2(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2, Object v2)
97      /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
98                    sa..contents = sb..contents & sa..csize = sb..csize"
99          modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
100         ensures "True" */
101     {
102         /*: assume "0 <= i1 & i1 <= sa..csize" */
103         sa.add_at(i1, v1);
104         /*: assume "0 <= i2 & i2 <= sa..csize" */
105         sa.add_at(i2, v2);
106         /*: assume "~((i1 < i2 & i2 <= sa..csize - 2 & (i2 + 1, v2) : sa..contents & 0
                <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & v1 = v2) | (i1 > i2 & (i1, v1)
                : sa..contents & 0 <= i1 & i1 < sa..csize))" */
107
108         /*: assume "0 <= i2 & i2 <= sb..csize" */
109         sb.add_at(i2, v2);
110         /*: assume "0 <= i1 & i1 <= sb..csize" */
111         sb.add_at(i1, v1);
112
113         /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
114     }
115
116     static void add_at_get_pre_s_3(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2)
117     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
118                   sa..contents = sb..contents & sa..csize = sb..csize"
119         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
120         ensures "True" */
121     {
122         /*: assume "(i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents) =
                ((i2, v) : sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2
                < sa..csize) | (i1 = i2 & i2 < sa..csize & (i1, v1) : sa..contents & 0 <= i1
                & i1 < sa..csize) | i1 > i2" */
123         /*: assume "0 <= i1 & i1 <= sa..csize" */
124         sa.add_at(i1, v1);
125         /*: assume "0 <= i2 & i2 < sa..csize" */
126         Object r2a = sa.get(i2);
127
128         Object r2b = sb.get(i2);
129         sb.add_at(i1, v1);
130
131         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
132     }
133
134     static void add_at_get_pre_c_3(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2)
135     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
136                   sa..contents = sb..contents & sa..csize = sb..csize"
137         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
138         ensures "True" */
```

```
    {
        /*: assume "~((i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents)
            = ((i2, v) : sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 &
            i2 < sa..csize) | (i1 = i2 & i2 < sa..csize & (i1, v1) : sa..contents & 0 <=
            i1 & i1 < sa..csize) | i1 > i2)" */
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.get(i2);
        /*: assume "0 <= i1 & i1 <= sb..csize" */
        sb.add_at(i1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void add_at_get_between_s_4(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "(i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa..contents) =
            ((i2 + 1, v) : sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2
            + 1 < sa..csize) | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, v1) :
            sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | i1 > i2" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        Object r2b = sb.get(i2);
        sb.add_at(i1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void add_at_get_between_c_4(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "~((i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa..contents)
            = ((i2 + 1, v) : sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 &
            i2 + 1 < sa..csize) | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, v1) :
            sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | i1 > i2)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.get(i2);
        /*: assume "0 <= i1 & i1 <= sb..csize" */
        sb.add_at(i1, v1);

```

```
189         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
190     }
191
192     static void add_at_get_post_s_5(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2)
193     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
194                     sa..contents = sb..contents & sa..csize = sb..csize"
195         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
196         ensures "True" */
197     {
198         /*: assume "0 <= i1 & i1 <= sa..csize" */
199         sa.add_at(i1, v1);
200         /*: assume "0 <= i2 & i2 < sa..csize" */
201         Object r2a = sa.get(i2);
202         /*: assume "(i1 < i2 & i2 < sa..csize - 1 & (i2 + 1, r2a) : sa..contents & 0 <=
                i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, v1)
                : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | i1 > i2" */
203
204         Object r2b = sb.get(i2);
205         sb.add_at(i1, v1);
206
207         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
208     }
209
210     static void add_at_get_post_c_5(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2)
211     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
212                     sa..contents = sb..contents & sa..csize = sb..csize"
213         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
214         ensures "True" */
215     {
216         /*: assume "0 <= i1 & i1 <= sa..csize" */
217         sa.add_at(i1, v1);
218         /*: assume "0 <= i2 & i2 < sa..csize" */
219         Object r2a = sa.get(i2);
220         /*: assume "~((i1 < i2 & i2 < sa..csize - 1 & (i2 + 1, r2a) : sa..contents & 0
                <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1,
                v1) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | i1 > i2)" */
221
222         /*: assume "0 <= i2 & i2 < sb..csize" */
223         Object r2b = sb.get(i2);
224         /*: assume "0 <= i1 & i1 <= sb..csize" */
225         sb.add_at(i1, v1);
226
227         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
228     }
229
230     static void add_at_indexOf_pre_s_6(ArrayList sa, ArrayList sb, int i1, Object v1,
            Object v2)
231     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
232                     sa..contents = sb..contents & sa..csize = sb..csize"
233         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
234         ensures "True" */
235     {
236         /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
                v2) | (EX i. (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) :
                sa..contents & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 <
                sa..csize & v1 = v2)" */
237         /*: assume "0 <= i1 & i1 <= sa..csize" */
238         sa.add_at(i1, v1);
```

```
        int r2a = sa.indexOf(v2);

        int r2b = sb.indexOf(v2);
        sb.add_at(i1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void add_at_indexOf_pre_c_6(ArrayList sa, ArrayList sb, int i1, Object v1,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
            v2) | (EX i. (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) :
            sa..contents & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 <
            sa..csize & v1 = v2))" */
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        int r2a = sa.indexOf(v2);

        int r2b = sb.indexOf(v2);
        /*: assume "0 <= i1 & i1 <= sb..csize" */
        sb.add_at(i1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void add_at_indexOf_between_c_7(ArrayList sa, ArrayList sb, int i1, Object
        v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "~(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
            (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) : sa..contents
            & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize &
            (i1 + 1, v2) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
        int r2a = sa.indexOf(v2);

        int r2b = sb.indexOf(v2);
        /*: assume "0 <= i1 & i1 <= sb..csize" */
        sb.add_at(i1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void add_at_indexOf_post_c_8(ArrayList sa, ArrayList sb, int i1, Object v1,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
```

```
290         sa.add_at(i1, v1);
291         int r2a = sa.indexOf(v2);
292         /*: assume "~(r2a < 0 | (0 <= r2a & r2a < i1) | (r2a = i1 & (i1 + 1, v2) :
                sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
293
294         int r2b = sb.indexOf(v2);
295         /*: assume "0 <= i1 & i1 <= sb..csize" */
296         sb.add_at(i1, v1);
297
298         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
299     }
300
301     static void add_at_lastIndexOf_pre_s_9(ArrayList sa, ArrayList sb, int i1, Object
            v1, Object v2)
302     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
303                 sa..contents = sb..contents & sa..csize = sb..csize"
304         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
305         ensures "True" */
306     {
307         /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
                v2) | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
                sa..contents & i1 <= i & i < sa..csize) & v1 ~= v2)" */
308         /*: assume "0 <= i1 & i1 <= sa..csize" */
309         sa.add_at(i1, v1);
310         int r2a = sa.lastIndexOf(v2);
311
312         int r2b = sb.lastIndexOf(v2);
313         sb.add_at(i1, v1);
314
315         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
316     }
317
318     static void add_at_remove_at_pre_s_12(ArrayList sa, ArrayList sb, int i1, Object
            v1, int i2)
319     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
320                 sa..contents = sb..contents & sa..csize = sb..csize"
321         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
322         ensures "True" */
323     {
324         /*: assume "(i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents) =
                ((i2, v) : sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2
                < sa..csize) | (sa..csize - 1 >= i1 & i1 = i2 & (i1, v1) : sa..contents & 0
                <= i1 & i1 < sa..csize) | (sa..csize - 1 >= i1 & i1 > i2 & (i1, v1) :
                sa..contents & 0 <= i1 & i1 < sa..csize)" */
325         /*: assume "0 <= i1 & i1 <= sa..csize" */
326         sa.add_at(i1, v1);
327         /*: assume "0 <= i2 & i2 < sa..csize" */
328         Object r2a = sa.remove_at(i2);
329
330         Object r2b = sb.remove_at(i2);
331         sb.add_at(i1, v1);
332
333         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
334     }
335
336     static void add_at_remove_at_pre_c_12(ArrayList sa, ArrayList sb, int i1, Object
            v1, int i2)
337     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
338                 sa..contents = sb..contents & sa..csize = sb..csize"
339         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
340         ensures "True" */
```

```
    {
        /*: assume "~((i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents)
            = ((i2, v) : sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 &
            i2 < sa..csize) | (sa..csize - 1 >= i1 & i1 = i2 & (i1, v1) : sa..contents &
            0 <= i1 & i1 < sa..csize) | (sa..csize - 1 >= i1 & i1 > i2 & (i1, v1) :
            sa..contents & 0 <= i1 & i1 < sa..csize))" */
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 <= sb..csize" */
        sb.add_at(i1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void add_at_remove_at_between_s_13(ArrayList sa, ArrayList sb, int i1,
        Object v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "(i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa..contents) =
            ((i2 + 1, v) : sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2
            + 1 < sa..csize) | (sa..csize - 2 >= i1 & i1 = i2 & (i1 + 1, v1) :
            sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 2 >= i1 & i1
            > i2 & (i1 + 1, v1) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);

        Object r2b = sb.remove_at(i2);
        sb.add_at(i1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void add_at_remove_at_between_c_13(ArrayList sa, ArrayList sb, int i1,
        Object v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "~((i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa..contents)
            = ((i2 + 1, v) : sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 &
            i2 + 1 < sa..csize) | (sa..csize - 2 >= i1 & i1 = i2 & (i1 + 1, v1) :
            sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 2 >= i1 & i1
            > i2 & (i1 + 1, v1) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 <= sb..csize" */
```

```
        sb.add_at(i1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void add_at_remove_at_post_s_14(ArrayList sa, ArrayList sb, int i1, Object
        v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);
        /*: assume "(i1 < i2 & i2 < sa..csize & (i2, r2a) : sa..contents & 0 <= i2 & i2
            < sa..csize) | (sa..csize - 1 >= i1 & i1 = i2 & (i1, v1) : sa..contents & 0
            <= i1 & i1 < sa..csize) | (sa..csize - 1 >= i1 & i1 > i2 & (i1, v1) :
            sa..contents & 0 <= i1 & i1 < sa..csize)" */

        Object r2b = sb.remove_at(i2);
        sb.add_at(i1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void add_at_remove_at_post_c_14(ArrayList sa, ArrayList sb, int i1, Object
        v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);
        /*: assume "~((i1 < i2 & i2 < sa..csize & (i2, r2a) : sa..contents & 0 <= i2 &
            i2 < sa..csize) | (sa..csize - 1 >= i1 & i1 = i2 & (i1, v1) : sa..contents &
            0 <= i1 & i1 < sa..csize) | (sa..csize - 1 >= i1 & i1 > i2 & (i1, v1) :
            sa..contents & 0 <= i1 & i1 < sa..csize))" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 <= sb..csize" */
        sb.add_at(i1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void add_at_remove_at_pre_s_15(ArrayList sa, ArrayList sb, int i1, Object
        v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "(i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents) =
            ((i2, v) : sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2
```

```
        < sa..csize) | (sa..csize - 1 >= i1 & i1 = i2 & (i1, v1) : sa..contents & 0
        <= i1 & i1 < sa..csize) | (sa..csize - 1 >= i1 & i1 > i2 & (i1, v1) :
        sa..contents & 0 <= i1 & i1 < sa..csize)" */
    /*: assume "0 <= i1 & i1 <= sa..csize" */
    sa.add_at(i1, v1);
    /*: assume "0 <= i2 & i2 < sa..csize" */
    sa.remove_at(i2);

    sb.remove_at(i2);
    sb.add_at(i1, v1);

    /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
}

static void add_at_remove_at_between_s_16(ArrayList sa, ArrayList sb, int i1,
    Object v1, int i2)
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
              sa..contents = sb..contents & sa..csize = sb..csize"
    modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
        "sb..msize"
    ensures "True" */
{
    /*: assume "0 <= i1 & i1 <= sa..csize" */
    sa.add_at(i1, v1);
    /*: assume "(i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa..contents) =
        ((i2 + 1, v) : sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2
        + 1 < sa..csize) | (sa..csize - 2 >= i1 & i1 = i2 & (i1 + 1, v1) :
        sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 2 >= i1 & i1
        > i2 & (i1 + 1, v1) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
    /*: assume "0 <= i2 & i2 < sa..csize" */
    sa.remove_at(i2);

    sb.remove_at(i2);
    sb.add_at(i1, v1);

    /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
}

static void add_at_remove_at_post_s_17(ArrayList sa, ArrayList sb, int i1, Object
    v1, int i2)
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
              sa..contents = sb..contents & sa..csize = sb..csize"
    modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
        "sb..msize"
    ensures "True" */
{
    /*: assume "0 <= i1 & i1 <= sa..csize" */
    sa.add_at(i1, v1);
    /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
    /*: ghost specvar sa__csize :: "int" = "sa..csize" */
    /*: assume "0 <= i2 & i2 < sa..csize" */
    sa.remove_at(i2);
    /*: assume "(i1 < i2 & i2 < sa..csize & (ALL v. ((i2, v) : sa__contents) =
        ((i2, v) : sa..contents)) & 0 <= i2 & i2 < sa__csize & 0 <= i2 & i2 <
        sa..csize) | (sa..csize - 1 >= i1 & i1 = i2 & (i1, v1) : sa..contents & 0 <=
        i1 & i1 < sa..csize) | (sa..csize - 1 >= i1 & i1 > i2 & (i1, v1) :
        sa..contents & 0 <= i1 & i1 < sa..csize)" */

    sb.remove_at(i2);
    sb.add_at(i1, v1);

    /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
}
```

```
488     static void add_at_remove_at_post_c_17(ArrayList sa, ArrayList sb, int i1, Object
            v1, int i2)
489     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
490                 sa..contents = sb..contents & sa..csize = sb..csize"
491         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
492         ensures "True" */
493     {
494         /*: assume "0 <= i1 & i1 <= sa..csize" */
495         sa.add_at(i1, v1);
496         /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
497         /*: ghost specvar sa__csize :: "int" = "sa..csize" */
498         /*: assume "0 <= i2 & i2 < sa..csize" */
499         sa.remove_at(i2);
500         /*: assume "~((i1 < i2 & i2 < sa..csize & (ALL v. ((i2, v) : sa__contents) =
                ((i2, v) : sa..contents)) & 0 <= i2 & i2 < sa__csize & 0 <= i2 & i2 <
                sa..csize) | (sa..csize - 1 >= i1 & i1 = i2 & (i1, v1) : sa..contents & 0 <=
                i1 & i1 < sa..csize) | (sa..csize - 1 >= i1 & i1 > i2 & (i1, v1) :
                sa..contents & 0 <= i1 & i1 < sa..csize))" */
501
502         /*: assume "0 <= i2 & i2 < sb..csize" */
503         sb.remove_at(i2);
504         /*: assume "0 <= i1 & i1 <= sb..csize" */
505         sb.add_at(i1, v1);
506
507         /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
508     }
509
510     static void add_at_set_pre_s_18(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2, Object v2)
511     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
512                 sa..contents = sb..contents & sa..csize = sb..csize"
513         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
514         ensures "True" */
515     {
516         /*: assume "(i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents) =
                ((i2, v) : sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) :
                sa..contents & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize)
                | (i1 = i2 & i2 < sa..csize & (i2, v1) : sa..contents & (i2, v2) :
                sa..contents & v1 = v2 & 0 <= i2 & i2 < sa..csize) | i1 > i2" */
517         /*: assume "0 <= i1 & i1 <= sa..csize" */
518         sa.add_at(i1, v1);
519         /*: assume "0 <= i2 & i2 < sa..csize" */
520         Object r2a = sa.set(i2, v2);
521
522         Object r2b = sb.set(i2, v2);
523         sb.add_at(i1, v1);
524
525         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
526     }
527
528     static void add_at_set_pre_c_18(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2, Object v2)
529     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
530                 sa..contents = sb..contents & sa..csize = sb..csize"
531         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
532         ensures "True" */
533     {
534         /*: assume "~((i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents)
                = ((i2, v) : sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) :
                sa..contents & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize)
                | (i1 = i2 & i2 < sa..csize & (i2, v1) : sa..contents & (i2, v2) :
                sa..contents & v1 = v2 & 0 <= i2 & i2 < sa..csize) | i1 > i2)" */
```

```
535         /*: assume "0 <= i1 & i1 <= sa..csize" */
536         sa.add_at(i1, v1);
537         /*: assume "0 <= i2 & i2 < sa..csize" */
538         Object r2a = sa.set(i2, v2);
539 
540         /*: assume "0 <= i2 & i2 < sb..csize" */
541         Object r2b = sb.set(i2, v2);
542         /*: assume "0 <= i1 & i1 <= sb..csize" */
543         sb.add_at(i1, v1);
544 
545         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
546     }
547 
548     static void add_at_set_between_s_19(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2, Object v2)
549     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
550                   sa..contents = sb..contents & sa..csize = sb..csize"
551         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
552         ensures "True" */
553     {
554         /*: assume "0 <= i1 & i1 <= sa..csize" */
555         sa.add_at(i1, v1);
556         /*: assume "(i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa..contents) =
                ((i2 + 1, v) : sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) :
                sa..contents & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize)
                | (i1 = i2 & i2 < sa..csize - 1 & (i2 + 1, v1) : sa..contents & (i2 + 1, v2)
                : sa..contents & v1 = v2 & 0 <= i2 + 1 & i2 + 1 < sa..csize) | i1 > i2" */
557         /*: assume "0 <= i2 & i2 < sa..csize" */
558         Object r2a = sa.set(i2, v2);
559 
560         Object r2b = sb.set(i2, v2);
561         sb.add_at(i1, v1);
562 
563         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
564     }
565 
566     static void add_at_set_between_c_19(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2, Object v2)
567     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
568                   sa..contents = sb..contents & sa..csize = sb..csize"
569         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
570         ensures "True" */
571     {
572         /*: assume "0 <= i1 & i1 <= sa..csize" */
573         sa.add_at(i1, v1);
574         /*: assume "~((i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa..contents)
                = ((i2 + 1, v) : sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) :
                sa..contents & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize)
                | (i1 = i2 & i2 < sa..csize - 1 & (i2 + 1, v1) : sa..contents & (i2 + 1, v2)
                : sa..contents & v1 = v2 & 0 <= i2 + 1 & i2 + 1 < sa..csize) | i1 > i2)" */
575         /*: assume "0 <= i2 & i2 < sa..csize" */
576         Object r2a = sa.set(i2, v2);
577 
578         /*: assume "0 <= i2 & i2 < sb..csize" */
579         Object r2b = sb.set(i2, v2);
580         /*: assume "0 <= i1 & i1 <= sb..csize" */
581         sb.add_at(i1, v1);
582 
583         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
584     }
585 
```

```
    static void add_at_set_post_s_20(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);
        /*: assume "(i1 < i2 & i2 < sa..csize - 1 & (i2 + 1, r2a) : sa..contents & r2a
            = v2 & (i2 + 1, v2) : sa..contents & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1
            = i2 & i2 < sa..csize - 1 & (i2 + 1, v1) : sa..contents & (i2 + 1, v2) :
            sa..contents & v1 = v2 & 0 <= i2 + 1 & i2 + 1 < sa..csize) | i1 > i2" */

        Object r2b = sb.set(i2, v2);
        sb.add_at(i1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void add_at_set_post_c_20(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);
        /*: assume "~((i1 < i2 & i2 < sa..csize - 1 & (i2 + 1, r2a) : sa..contents &
            r2a = v2 & (i2 + 1, v2) : sa..contents & 0 <= i2 + 1 & i2 + 1 < sa..csize) |
            (i1 = i2 & i2 < sa..csize - 1 & (i2 + 1, v1) : sa..contents & (i2 + 1, v2) :
            sa..contents & v1 = v2 & 0 <= i2 + 1 & i2 + 1 < sa..csize) | i1 > i2)" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 <= sb..csize" */
        sb.add_at(i1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void add_at_set_pre_s_21(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "(i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents) =
            ((i2, v) : sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) :
            sa..contents & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize)
            | (i1 = i2 & i2 < sa..csize & (i2, v1) : sa..contents & (i2, v2) :
            sa..contents & v1 = v2 & 0 <= i2 & i2 < sa..csize) | i1 > i2" */
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
```

```
634         sa.set(i2, v2);
635
636         sb.set(i2, v2);
637         sb.add_at(i1, v1);
638
639         /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
640     }
641
642     static void add_at_set_between_s_22(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2, Object v2)
643     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
644                 sa..contents = sb..contents & sa..csize = sb..csize"
645         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
646         ensures "True" */
647     {
648         /*: assume "0 <= i1 & i1 <= sa..csize" */
649         sa.add_at(i1, v1);
650         /*: assume "(i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa..contents) =
                ((i2 + 1, v) : sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) :
                sa..contents & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize)
                | (i1 = i2 & i2 < sa..csize - 1 & (i2 + 1, v1) : sa..contents & (i2 + 1, v2)
                : sa..contents & v1 = v2 & 0 <= i2 + 1 & i2 + 1 < sa..csize) | i1 > i2" */
651         /*: assume "0 <= i2 & i2 < sa..csize" */
652         sa.set(i2, v2);
653
654         sb.set(i2, v2);
655         sb.add_at(i1, v1);
656
657         /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
658     }
659
660     static void add_at_set_between_c_22(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2, Object v2)
661     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
662                 sa..contents = sb..contents & sa..csize = sb..csize"
663         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
664         ensures "True" */
665     {
666         /*: assume "0 <= i1 & i1 <= sa..csize" */
667         sa.add_at(i1, v1);
668         /*: assume "~((i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa..contents)
                = ((i2 + 1, v) : sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) :
                sa..contents & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize)
                | (i1 = i2 & i2 < sa..csize - 1 & (i2 + 1, v1) : sa..contents & (i2 + 1, v2)
                : sa..contents & v1 = v2 & 0 <= i2 + 1 & i2 + 1 < sa..csize) | i1 > i2)" */
669         /*: assume "0 <= i2 & i2 < sa..csize" */
670         sa.set(i2, v2);
671
672         /*: assume "0 <= i2 & i2 < sb..csize" */
673         sb.set(i2, v2);
674         /*: assume "0 <= i1 & i1 <= sb..csize" */
675         sb.add_at(i1, v1);
676
677         /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
678     }
679
680     static void add_at_set_post_s_23(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
681     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
682                 sa..contents = sb..contents & sa..csize = sb..csize"
683         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
684         ensures "True" */
```

```
685     {
686         /*: assume "0 <= i1 & i1 <= sa..csize" */
687         sa.add_at(i1, v1);
688         /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
689         /*: ghost specvar sa__csize :: "int" = "sa..csize" */
690         /*: assume "0 <= i2 & i2 < sa..csize" */
691         sa.set(i2, v2);
692         /*: assume "(i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa__contents) =
                ((i2 + 1, v) : sa..contents)) & (i2, v2) : sa__contents & (i2 + 1, v2) :
                sa..contents & 0 <= i2 & i2 < sa__csize & 0 <= i2 + 1 & i2 + 1 < sa..csize)
                | (i1 = i2 & i2 < sa..csize - 1 & (i2 + 1, v1) : sa..contents & (i2 + 1, v2)
                : sa..contents & v1 = v2 & 0 <= i2 + 1 & i2 + 1 < sa..csize) | i1 > i2" */
693
694         sb.set(i2, v2);
695         sb.add_at(i1, v1);
696
697         /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
698     }
699
700     static void add_at_set_post_c_23(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2, Object v2)
701     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
702                   sa..contents = sb..contents & sa..csize = sb..csize"
703         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
704         ensures "True" */
705     {
706         /*: assume "0 <= i1 & i1 <= sa..csize" */
707         sa.add_at(i1, v1);
708         /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
709         /*: ghost specvar sa__csize :: "int" = "sa..csize" */
710         /*: assume "0 <= i2 & i2 < sa..csize" */
711         sa.set(i2, v2);
712         /*: assume "~((i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa__contents)
                = ((i2 + 1, v) : sa..contents)) & (i2, v2) : sa__contents & (i2 + 1, v2) :
                sa..contents & 0 <= i2 & i2 < sa__csize & 0 <= i2 + 1 & i2 + 1 < sa..csize)
                | (i1 = i2 & i2 < sa..csize - 1 & (i2 + 1, v1) : sa..contents & (i2 + 1, v2)
                : sa..contents & v1 = v2 & 0 <= i2 + 1 & i2 + 1 < sa..csize) | i1 > i2)" */
713
714         /*: assume "0 <= i2 & i2 < sb..csize" */
715         sb.set(i2, v2);
716         /*: assume "0 <= i1 & i1 <= sb..csize" */
717         sb.add_at(i1, v1);
718
719         /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
720     }
721
722     static void add_at_size_pre_s_24(ArrayList sa, ArrayList sb, int i1, Object v1)
723     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
724                   sa..contents = sb..contents & sa..csize = sb..csize"
725         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
726         ensures "True" */
727     {
728         /*: assume "False" */
729         /*: assume "0 <= i1 & i1 <= sa..csize" */
730         sa.add_at(i1, v1);
731         int r2a = sa.size();
732
733         int r2b = sb.size();
734         sb.add_at(i1, v1);
735
736         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
737     }
738
```

```
    static void add_at_size_pre_c_24(ArrayList sa, ArrayList sb, int i1, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "~(False)" */
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        int r2a = sa.size();

        int r2b = sb.size();
        /*: assume "0 <= i1 & i1 <= sb..csize" */
        sb.add_at(i1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void add_at_size_between_s_25(ArrayList sa, ArrayList sb, int i1, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "False" */
        int r2a = sa.size();

        int r2b = sb.size();
        sb.add_at(i1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void add_at_size_between_c_25(ArrayList sa, ArrayList sb, int i1, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "~(False)" */
        int r2a = sa.size();

        int r2b = sb.size();
        /*: assume "0 <= i1 & i1 <= sb..csize" */
        sb.add_at(i1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void add_at_size_post_s_26(ArrayList sa, ArrayList sb, int i1, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
```

```
798         /*: assume "0 <= i1 & i1 <= sa..csize" */
799         sa.add_at(i1, v1);
800         int r2a = sa.size();
801         /*: assume "False" */
802 
803         int r2b = sb.size();
804         sb.add_at(i1, v1);
805 
806         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
807     }
808 
809     static void add_at_size_post_c_26(ArrayList sa, ArrayList sb, int i1, Object v1)
810     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
811                 sa..contents = sb..contents & sa..csize = sb..csize"
812         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
813         ensures "True" */
814     {
815         /*: assume "0 <= i1 & i1 <= sa..csize" */
816         sa.add_at(i1, v1);
817         int r2a = sa.size();
818         /*: assume "~(False)" */
819 
820         int r2b = sb.size();
821         /*: assume "0 <= i1 & i1 <= sb..csize" */
822         sb.add_at(i1, v1);
823 
824         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
825     }
826 
827     static void get_add_at_pre_s_27(ArrayList sa, ArrayList sb, int i1, int i2, Object
            v2)
828     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
829                 sa..contents = sb..contents & sa..csize = sb..csize"
830         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
831         ensures "True" */
832     {
833         /*: assume "i1 < i2 | (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 <
                sa..csize) | (i1 > i2 & (ALL v. ((i1 - 1, v) : sa..contents) = ((i1, v) :
                sa..contents)) & 0 <= i1 - 1 & i1 - 1 < sa..csize & 0 <= i1 & i1 <
                sa..csize)" */
834         /*: assume "0 <= i1 & i1 < sa..csize" */
835         Object r1a = sa.get(i1);
836         /*: assume "0 <= i2 & i2 <= sa..csize" */
837         sa.add_at(i2, v2);
838 
839         sb.add_at(i2, v2);
840         Object r1b = sb.get(i1);
841 
842         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
843     }
844 
845     static void get_add_at_pre_c_27(ArrayList sa, ArrayList sb, int i1, int i2, Object
            v2)
846     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
847                 sa..contents = sb..contents & sa..csize = sb..csize"
848         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
849         ensures "True" */
850     {
851         /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 <
                sa..csize) | (i1 > i2 & (ALL v. ((i1 - 1, v) : sa..contents) = ((i1, v) :
```

```
            sa..contents)) & 0 <= i1 - 1 & i1 - 1 < sa..csize & 0 <= i1 & i1 <
            sa..csize))" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        /*: assume "0 <= i2 & i2 <= sb..csize" */
        sb.add_at(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void get_add_at_between_s_28(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "i1 < i2 | (i1 = i2 & r1a = v2) | (i1 > i2 & (i1 - 1, r1a) :
            sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize)" */
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        sb.add_at(i2, v2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void get_add_at_between_c_28(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v2) | (i1 > i2 & (i1 - 1, r1a) :
            sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize))" */
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        /*: assume "0 <= i2 & i2 <= sb..csize" */
        sb.add_at(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void get_add_at_post_s_29(ArrayList sa, ArrayList sb, int i1, int i2, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
```

```
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);
        /*: assume "i1 < i2 | (i1 = i2 & r1a = v2) | (i1 > i2 & (i1, r1a) :
            sa..contents & 0 <= i1 & i1 < sa..csize)" */

        sb.add_at(i2, v2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void get_add_at_post_c_29(ArrayList sa, ArrayList sb, int i1, int i2, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);
        /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v2) | (i1 > i2 & (i1, r1a) :
            sa..contents & 0 <= i1 & i1 < sa..csize))" */

        /*: assume "0 <= i2 & i2 <= sb..csize" */
        sb.add_at(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void get_get_pre_s_30(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "True" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        Object r2b = sb.get(i2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void get_get_pre_c_30(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "~(True)" */
```

```
964         /*: assume "0 <= i1 & i1 < sa..csize" */
965         Object r1a = sa.get(i1);
966         /*: assume "0 <= i2 & i2 < sa..csize" */
967         Object r2a = sa.get(i2);
968 
969         /*: assume "0 <= i2 & i2 < sb..csize" */
970         Object r2b = sb.get(i2);
971         /*: assume "0 <= i1 & i1 < sb..csize" */
972         Object r1b = sb.get(i1);
973 
974         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
975     }
976 
977     static void get_get_between_s_31(ArrayList sa, ArrayList sb, int i1, int i2)
978     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
979                   sa..contents = sb..contents & sa..csize = sb..csize"
980         ensures "True" */
981     {
982         /*: assume "0 <= i1 & i1 < sa..csize" */
983         Object r1a = sa.get(i1);
984         /*: assume "True" */
985         /*: assume "0 <= i2 & i2 < sa..csize" */
986         Object r2a = sa.get(i2);
987 
988         Object r2b = sb.get(i2);
989         Object r1b = sb.get(i1);
990 
991         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
992     }
993 
994     static void get_get_between_c_31(ArrayList sa, ArrayList sb, int i1, int i2)
995     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
996                   sa..contents = sb..contents & sa..csize = sb..csize"
997         ensures "True" */
998     {
999         /*: assume "0 <= i1 & i1 < sa..csize" */
1000        Object r1a = sa.get(i1);
1001        /*: assume "~(True)" */
1002        /*: assume "0 <= i2 & i2 < sa..csize" */
1003        Object r2a = sa.get(i2);
1004 
1005        /*: assume "0 <= i2 & i2 < sb..csize" */
1006        Object r2b = sb.get(i2);
1007        /*: assume "0 <= i1 & i1 < sb..csize" */
1008        Object r1b = sb.get(i1);
1009 
1010        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
1011    }
1012 
1013    static void get_get_post_s_32(ArrayList sa, ArrayList sb, int i1, int i2)
1014    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1015                  sa..contents = sb..contents & sa..csize = sb..csize"
1016        ensures "True" */
1017    {
1018        /*: assume "0 <= i1 & i1 < sa..csize" */
1019        Object r1a = sa.get(i1);
1020        /*: assume "0 <= i2 & i2 < sa..csize" */
1021        Object r2a = sa.get(i2);
1022        /*: assume "True" */
1023 
1024        Object r2b = sb.get(i2);
1025        Object r1b = sb.get(i1);
```

```
1026
1027        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
1028    }
1029
1030    static void get_get_post_c_32(ArrayList sa, ArrayList sb, int i1, int i2)
1031    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1032                  sa..contents = sb..contents & sa..csize = sb..csize"
1033        ensures "True" */
1034    {
1035        /*: assume "0 <= i1 & i1 < sa..csize" */
1036        Object r1a = sa.get(i1);
1037        /*: assume "0 <= i2 & i2 < sa..csize" */
1038        Object r2a = sa.get(i2);
1039        /*: assume "~(True)" */
1040
1041        /*: assume "0 <= i2 & i2 < sb..csize" */
1042        Object r2b = sb.get(i2);
1043        /*: assume "0 <= i1 & i1 < sb..csize" */
1044        Object r1b = sb.get(i1);
1045
1046        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
1047    }
1048
1049    static void get_indexOf_pre_s_33(ArrayList sa, ArrayList sb, int i1, Object v2)
1050    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1051                  sa..contents = sb..contents & sa..csize = sb..csize"
1052        ensures "True" */
1053    {
1054        /*: assume "True" */
1055        /*: assume "0 <= i1 & i1 < sa..csize" */
1056        Object r1a = sa.get(i1);
1057        int r2a = sa.indexOf(v2);
1058
1059        int r2b = sb.indexOf(v2);
1060        Object r1b = sb.get(i1);
1061
1062        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
1063    }
1064
1065    static void get_indexOf_pre_c_33(ArrayList sa, ArrayList sb, int i1, Object v2)
1066    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1067                  sa..contents = sb..contents & sa..csize = sb..csize"
1068        ensures "True" */
1069    {
1070        /*: assume "~(True)" */
1071        /*: assume "0 <= i1 & i1 < sa..csize" */
1072        Object r1a = sa.get(i1);
1073        int r2a = sa.indexOf(v2);
1074
1075        int r2b = sb.indexOf(v2);
1076        /*: assume "0 <= i1 & i1 < sb..csize" */
1077        Object r1b = sb.get(i1);
1078
1079        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
1080    }
1081
1082    static void get_indexOf_between_s_34(ArrayList sa, ArrayList sb, int i1, Object v2)
1083    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1084                  sa..contents = sb..contents & sa..csize = sb..csize"
1085        ensures "True" */
1086    {
```

```
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "True" */
        int r2a = sa.indexOf(v2);

        int r2b = sb.indexOf(v2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void get_indexOf_between_c_34(ArrayList sa, ArrayList sb, int i1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "~(True)" */
        int r2a = sa.indexOf(v2);

        int r2b = sb.indexOf(v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void get_indexOf_post_s_35(ArrayList sa, ArrayList sb, int i1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        int r2a = sa.indexOf(v2);
        /*: assume "True" */

        int r2b = sb.indexOf(v2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void get_indexOf_post_c_35(ArrayList sa, ArrayList sb, int i1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        int r2a = sa.indexOf(v2);
        /*: assume "~(True)" */

        int r2b = sb.indexOf(v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }
```

```
1148    static void get_lastIndexOf_pre_s_36(ArrayList sa, ArrayList sb, int i1, Object v2)
1149    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1150                  sa..contents = sb..contents & sa..csize = sb..csize"
1151        ensures "True" */
1152    {
1153        /*: assume "True" */
1154        /*: assume "0 <= i1 & i1 < sa..csize" */
1155        Object r1a = sa.get(i1);
1156        int r2a = sa.lastIndexOf(v2);
1157
1158        int r2b = sb.lastIndexOf(v2);
1159        Object r1b = sb.get(i1);
1160
1161        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
1162    }
1163
1164    static void get_lastIndexOf_pre_c_36(ArrayList sa, ArrayList sb, int i1, Object v2)
1165    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1166                  sa..contents = sb..contents & sa..csize = sb..csize"
1167        ensures "True" */
1168    {
1169        /*: assume "~(True)" */
1170        /*: assume "0 <= i1 & i1 < sa..csize" */
1171        Object r1a = sa.get(i1);
1172        int r2a = sa.lastIndexOf(v2);
1173
1174        int r2b = sb.lastIndexOf(v2);
1175        /*: assume "0 <= i1 & i1 < sb..csize" */
1176        Object r1b = sb.get(i1);
1177
1178        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
1179    }
1180
1181    static void get_lastIndexOf_between_s_37(ArrayList sa, ArrayList sb, int i1, Object
            v2)
1182    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1183                  sa..contents = sb..contents & sa..csize = sb..csize"
1184        ensures "True" */
1185    {
1186        /*: assume "0 <= i1 & i1 < sa..csize" */
1187        Object r1a = sa.get(i1);
1188        /*: assume "True" */
1189        int r2a = sa.lastIndexOf(v2);
1190
1191        int r2b = sb.lastIndexOf(v2);
1192        Object r1b = sb.get(i1);
1193
1194        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
1195    }
1196
1197    static void get_lastIndexOf_between_c_37(ArrayList sa, ArrayList sb, int i1, Object
            v2)
1198    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1199                  sa..contents = sb..contents & sa..csize = sb..csize"
1200        ensures "True" */
1201    {
1202        /*: assume "0 <= i1 & i1 < sa..csize" */
1203        Object r1a = sa.get(i1);
1204        /*: assume "~(True)" */
1205        int r2a = sa.lastIndexOf(v2);
1206
1207        int r2b = sb.lastIndexOf(v2);
```

```
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void get_lastIndexOf_post_s_38(ArrayList sa, ArrayList sb, int i1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        int r2a = sa.lastIndexOf(v2);
        /*: assume "True" */

        int r2b = sb.lastIndexOf(v2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void get_lastIndexOf_post_c_38(ArrayList sa, ArrayList sb, int i1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        int r2a = sa.lastIndexOf(v2);
        /*: assume "~(True)" */

        int r2b = sb.lastIndexOf(v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void get_remove_at_pre_s_39(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "i1 < i2 | (sa..csize - 1 > i1 & i1 = i2 & (ALL v. ((i1, v) :
            sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
            <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v.
            ((i1, v) : sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 <
            sa..csize & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);

        Object r2b = sb.remove_at(i2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }
```

```
    static void get_remove_at_pre_c_39(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~(i1 < i2 | (sa..csize - 1 > i1 & i1 = i2 & (ALL v. ((i1, v) :
            sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
            <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v.
            ((i1, v) : sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 <
            sa..csize & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void get_remove_at_between_s_40(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "i1 < i2 | (sa..csize - 1 > i1 & i1 = i2 & (i1 + 1, r1a) :
            sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
            > i2 & (i1 + 1, r1a) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);

        Object r2b = sb.remove_at(i2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void get_remove_at_between_c_40(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "~(i1 < i2 | (sa..csize - 1 > i1 & i1 = i2 & (i1 + 1, r1a) :
            sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
            > i2 & (i1 + 1, r1a) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

```

```
        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void get_remove_at_post_s_41(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);
        /*: assume "i1 < i2 | (sa..csize > i1 & i1 = i2 & (i1, r1a) : sa..contents & 0
            <= i1 & i1 < sa..csize) | (sa..csize > i1 & i1 > i2 & (i1, r1a) :
            sa..contents & 0 <= i1 & i1 < sa..csize)" */

        Object r2b = sb.remove_at(i2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void get_remove_at_post_c_41(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);
        /*: assume "~(i1 < i2 | (sa..csize > i1 & i1 = i2 & (i1, r1a) : sa..contents &
            0 <= i1 & i1 < sa..csize) | (sa..csize > i1 & i1 > i2 & (i1, r1a) :
            sa..contents & 0 <= i1 & i1 < sa..csize))" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void get_remove_at_pre_s_42(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "i1 < i2 | (sa..csize - 1 > i1 & i1 = i2 & (ALL v. ((i1, v) :
            sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
            <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v.
            ((i1, v) : sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 <
            sa..csize & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        sb.remove_at(i2);
```

```
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void get_remove_at_pre_c_42(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~(i1 < i2 | (sa..csize - 1 > i1 & i1 = i2 & (ALL v. ((i1, v) :
            sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
            <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v.
            ((i1, v) : sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 <
            sa..csize & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void get_remove_at_between_s_43(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "i1 < i2 | (sa..csize - 1 > i1 & i1 = i2 & (i1 + 1, r1a) :
            sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
            > i2 & (i1 + 1, r1a) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        sb.remove_at(i2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void get_remove_at_between_c_43(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "~(i1 < i2 | (sa..csize - 1 > i1 & i1 = i2 & (i1 + 1, r1a) :
            sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
            > i2 & (i1 + 1, r1a) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
```

```
        sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void get_remove_at_post_s_44(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);
        /*: assume "i1 < i2 | (sa..csize > i1 & i1 = i2 & (i1, r1a) : sa..contents & 0
            <= i1 & i1 < sa..csize) | (sa..csize > i1 & i1 > i2 & (i1, r1a) :
            sa..contents & 0 <= i1 & i1 < sa..csize)" */

        sb.remove_at(i2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void get_remove_at_post_c_44(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);
        /*: assume "~(i1 < i2 | (sa..csize > i1 & i1 = i2 & (i1, r1a) : sa..contents &
            0 <= i1 & i1 < sa..csize) | (sa..csize > i1 & i1 > i2 & (i1, r1a) :
            sa..contents & 0 <= i1 & i1 < sa..csize))" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void get_set_pre_s_45(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "i1 < i2 | (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 <
            sa..csize) | i1 > i2" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        Object r2b = sb.set(i2, v2);
```

```
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void get_set_pre_c_45(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 <
            sa..csize) | i1 > i2)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void get_set_between_s_46(ArrayList sa, ArrayList sb, int i1, int i2, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        Object r2b = sb.set(i2, v2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void get_set_between_c_46(ArrayList sa, ArrayList sb, int i1, int i2, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);
```

```
1547
1548        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
1549    }
1550
1551    static void get_set_post_s_47(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
1552    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1553                  sa..contents = sb..contents & sa..csize = sb..csize"
1554        modifies "sa..contents", "sb..contents"
1555        ensures "True" */
1556    {
1557        /*: assume "0 <= i1 & i1 < sa..csize" */
1558        Object r1a = sa.get(i1);
1559        /*: assume "0 <= i2 & i2 < sa..csize" */
1560        Object r2a = sa.set(i2, v2);
1561        /*: assume "i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2" */
1562
1563        Object r2b = sb.set(i2, v2);
1564        Object r1b = sb.get(i1);
1565
1566        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
1567    }
1568
1569    static void get_set_post_c_47(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
1570    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1571                  sa..contents = sb..contents & sa..csize = sb..csize"
1572        modifies "sa..contents", "sb..contents"
1573        ensures "True" */
1574    {
1575        /*: assume "0 <= i1 & i1 < sa..csize" */
1576        Object r1a = sa.get(i1);
1577        /*: assume "0 <= i2 & i2 < sa..csize" */
1578        Object r2a = sa.set(i2, v2);
1579        /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2)" */
1580
1581        /*: assume "0 <= i2 & i2 < sb..csize" */
1582        Object r2b = sb.set(i2, v2);
1583        /*: assume "0 <= i1 & i1 < sb..csize" */
1584        Object r1b = sb.get(i1);
1585
1586        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
1587    }
1588
1589    static void get_set_pre_s_48(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
1590    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1591                  sa..contents = sb..contents & sa..csize = sb..csize"
1592        modifies "sa..contents", "sb..contents"
1593        ensures "True" */
1594    {
1595        /*: assume "i1 < i2 | (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 <
                sa..csize) | i1 > i2" */
1596        /*: assume "0 <= i1 & i1 < sa..csize" */
1597        Object r1a = sa.get(i1);
1598        /*: assume "0 <= i2 & i2 < sa..csize" */
1599        sa.set(i2, v2);
1600
1601        sb.set(i2, v2);
1602        Object r1b = sb.get(i1);
1603
1604        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
1605    }
1606
1607    static void get_set_pre_c_48(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
```

```
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 <
            sa..csize) | i1 > i2)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void get_set_between_s_49(ArrayList sa, ArrayList sb, int i1, int i2, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        sb.set(i2, v2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void get_set_between_c_49(ArrayList sa, ArrayList sb, int i1, int i2, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void get_set_post_s_50(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
```

```
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);
        /*: assume "i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2" */

        sb.set(i2, v2);
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void get_set_post_c_50(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);
        /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2)" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void get_size_pre_s_51(ArrayList sa, ArrayList sb, int i1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "True" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        int r2a = sa.size();

        int r2b = sb.size();
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void get_size_pre_c_51(ArrayList sa, ArrayList sb, int i1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "~(True)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        int r2a = sa.size();

        int r2b = sb.size();
        /*: assume "0 <= i1 & i1 < sb..csize" */
```

```
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void get_size_between_s_52(ArrayList sa, ArrayList sb, int i1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "True" */
        int r2a = sa.size();

        int r2b = sb.size();
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void get_size_between_c_52(ArrayList sa, ArrayList sb, int i1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        /*: assume "~(True)" */
        int r2a = sa.size();

        int r2b = sb.size();
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void get_size_post_s_53(ArrayList sa, ArrayList sb, int i1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
        int r2a = sa.size();
        /*: assume "True" */

        int r2b = sb.size();
        Object r1b = sb.get(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void get_size_post_c_53(ArrayList sa, ArrayList sb, int i1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.get(i1);
```

```
        int r2a = sa.size();
        /*: assume "~(True)" */

        int r2b = sb.size();
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.get(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void indexOf_add_at_pre_s_54(ArrayList sa, ArrayList sb, Object v1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "(~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
            v2) | (EX i. (i, v1) : sa..contents & 0 <= i & i < i2) | (~(EX i. (i, v1) :
            sa..contents & 0 <= i & i < i2) & (i2, v1) : sa..contents & 0 <= i2 & i2 <
            sa..csize & v1 = v2)" */
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        sb.add_at(i2, v2);
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void indexOf_add_at_pre_c_54(ArrayList sa, ArrayList sb, Object v1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "~((~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
            v2) | (EX i. (i, v1) : sa..contents & 0 <= i & i < i2) | (~(EX i. (i, v1) :
            sa..contents & 0 <= i & i < i2) & (i2, v1) : sa..contents & 0 <= i2 & i2 <
            sa..csize & v1 = v2))" */
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        /*: assume "0 <= i2 & i2 <= sb..csize" */
        sb.add_at(i2, v2);
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void indexOf_add_at_between_s_55(ArrayList sa, ArrayList sb, Object v1, int
        i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
```

```
        int r1a = sa.indexOf(v1);
        /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 =
            v2)" */
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        sb.add_at(i2, v2);
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void indexOf_add_at_between_c_55(ArrayList sa, ArrayList sb, Object v1, int
        i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 =
            v2))" */
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        /*: assume "0 <= i2 & i2 <= sb..csize" */
        sb.add_at(i2, v2);
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void indexOf_add_at_post_s_56(ArrayList sa, ArrayList sb, Object v1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);
        /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 =
            v2)" */

        sb.add_at(i2, v2);
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void indexOf_add_at_post_c_56(ArrayList sa, ArrayList sb, Object v1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);
```

```
        /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 =
            v2))" */

        /*: assume "0 <= i2 & i2 <= sb..csize" */
        sb.add_at(i2, v2);
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void indexOf_get_pre_s_57(ArrayList sa, ArrayList sb, Object v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "True" */
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        Object r2b = sb.get(i2);
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void indexOf_get_pre_c_57(ArrayList sa, ArrayList sb, Object v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "~(True)" */
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.get(i2);
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void indexOf_get_between_s_58(ArrayList sa, ArrayList sb, Object v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "True" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        Object r2b = sb.get(i2);
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void indexOf_get_between_c_58(ArrayList sa, ArrayList sb, Object v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
```

```
                sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "~(True)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.get(i2);
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void indexOf_get_post_s_59(ArrayList sa, ArrayList sb, Object v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);
        /*: assume "True" */

        Object r2b = sb.get(i2);
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void indexOf_get_post_c_59(ArrayList sa, ArrayList sb, Object v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);
        /*: assume "~(True)" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.get(i2);
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void indexOf_indexOf_pre_s_60(ArrayList sa, ArrayList sb, Object v1, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "True" */
        int r1a = sa.indexOf(v1);
        int r2a = sa.indexOf(v2);

        int r2b = sb.indexOf(v2);
        int r1b = sb.indexOf(v1);

```

```
        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void indexOf_indexOf_pre_c_60(ArrayList sa, ArrayList sb, Object v1, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "~(True)" */
        int r1a = sa.indexOf(v1);
        int r2a = sa.indexOf(v2);

        int r2b = sb.indexOf(v2);
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void indexOf_indexOf_between_s_61(ArrayList sa, ArrayList sb, Object v1,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "True" */
        int r2a = sa.indexOf(v2);

        int r2b = sb.indexOf(v2);
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void indexOf_indexOf_between_c_61(ArrayList sa, ArrayList sb, Object v1,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "~(True)" */
        int r2a = sa.indexOf(v2);

        int r2b = sb.indexOf(v2);
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void indexOf_indexOf_post_s_62(ArrayList sa, ArrayList sb, Object v1, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        int r2a = sa.indexOf(v2);
        /*: assume "True" */

```

```
        int r2b = sb.indexOf(v2);
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void indexOf_indexOf_post_c_62(ArrayList sa, ArrayList sb, Object v1, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        int r2a = sa.indexOf(v2);
        /*: assume "~(True)" */

        int r2b = sb.indexOf(v2);
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void indexOf_lastIndexOf_pre_s_63(ArrayList sa, ArrayList sb, Object v1,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "True" */
        int r1a = sa.indexOf(v1);
        int r2a = sa.lastIndexOf(v2);

        int r2b = sb.lastIndexOf(v2);
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void indexOf_lastIndexOf_pre_c_63(ArrayList sa, ArrayList sb, Object v1,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "~(True)" */
        int r1a = sa.indexOf(v1);
        int r2a = sa.lastIndexOf(v2);

        int r2b = sb.lastIndexOf(v2);
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void indexOf_lastIndexOf_between_s_64(ArrayList sa, ArrayList sb, Object v1,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
```

```
2132        /*: assume "True" */
2133        int r2a = sa.lastIndexOf(v2);
2134
2135        int r2b = sb.lastIndexOf(v2);
2136        int r1b = sb.indexOf(v1);
2137
2138        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
2139    }
2140
2141    static void indexOf_lastIndexOf_between_c_64(ArrayList sa, ArrayList sb, Object v1,
            Object v2)
2142    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2143                  sa..contents = sb..contents & sa..csize = sb..csize"
2144        ensures "True" */
2145    {
2146        int r1a = sa.indexOf(v1);
2147        /*: assume "~(True)" */
2148        int r2a = sa.lastIndexOf(v2);
2149
2150        int r2b = sb.lastIndexOf(v2);
2151        int r1b = sb.indexOf(v1);
2152
2153        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
2154    }
2155
2156    static void indexOf_lastIndexOf_post_s_65(ArrayList sa, ArrayList sb, Object v1,
            Object v2)
2157    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2158                  sa..contents = sb..contents & sa..csize = sb..csize"
2159        ensures "True" */
2160    {
2161        int r1a = sa.indexOf(v1);
2162        int r2a = sa.lastIndexOf(v2);
2163        /*: assume "True" */
2164
2165        int r2b = sb.lastIndexOf(v2);
2166        int r1b = sb.indexOf(v1);
2167
2168        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
2169    }
2170
2171    static void indexOf_lastIndexOf_post_c_65(ArrayList sa, ArrayList sb, Object v1,
            Object v2)
2172    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2173                  sa..contents = sb..contents & sa..csize = sb..csize"
2174        ensures "True" */
2175    {
2176        int r1a = sa.indexOf(v1);
2177        int r2a = sa.lastIndexOf(v2);
2178        /*: assume "~(True)" */
2179
2180        int r2b = sb.lastIndexOf(v2);
2181        int r1b = sb.indexOf(v1);
2182
2183        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
2184    }
2185
2186    static void indexOf_remove_at_post_c_68(ArrayList sa, ArrayList sb, Object v1, int
            i2)
2187    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2188                  sa..contents = sb..contents & sa..csize = sb..csize"
```

```
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);
        /*: assume "~(r1a < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize &
            (i2, v1) : sa..contents & 0 <= i2 & i2 < sa..csize))" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.remove_at(i2);
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void indexOf_set_pre_s_72(ArrayList sa, ArrayList sb, Object v1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "(~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
            v2) | (EX i. (i, v1) : sa..contents & 0 <= i & i < i2) | (~(EX i. (i, v1) :
            sa..contents & 0 <= i & i < i2) & (i2, v1) : sa..contents & v1 = v2) | (~(EX
            i. (i, v1) : sa..contents & 0 <= i & i <= i2) & (EX i. (i, v1) :
            sa..contents & i2 < i & i < sa..csize) & v1 ~= v2)" */
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        Object r2b = sb.set(i2, v2);
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void indexOf_set_pre_c_72(ArrayList sa, ArrayList sb, Object v1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "~((~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
            v2) | (EX i. (i, v1) : sa..contents & 0 <= i & i < i2) | (~(EX i. (i, v1) :
            sa..contents & 0 <= i & i < i2) & (i2, v1) : sa..contents & v1 = v2) | (~(EX
            i. (i, v1) : sa..contents & 0 <= i & i <= i2) & (EX i. (i, v1) :
            sa..contents & i2 < i & i < sa..csize) & v1 ~= v2))" */
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.set(i2, v2);
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }
```

```
    static void indexOf_set_between_s_73(ArrayList sa, ArrayList sb, Object v1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 = v2)
            | (r1a > i2 & v1 ~= v2)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        Object r2b = sb.set(i2, v2);
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void indexOf_set_between_c_73(ArrayList sa, ArrayList sb, Object v1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 =
            v2) | (r1a > i2 & v1 ~= v2))" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.set(i2, v2);
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void indexOf_set_post_s_74(ArrayList sa, ArrayList sb, Object v1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);
        /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 = v2)
            | (r1a > i2 & v1 ~= v2)" */

        Object r2b = sb.set(i2, v2);
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void indexOf_set_post_c_74(ArrayList sa, ArrayList sb, Object v1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
```

```
2294        modifies "sa..contents", "sb..contents"
2295        ensures "True" */
2296    {
2297        int r1a = sa.indexOf(v1);
2298        /*: assume "0 <= i2 & i2 < sa..csize" */
2299        Object r2a = sa.set(i2, v2);
2300        /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 =
                v2) | (r1a > i2 & v1 ~= v2))" */
2301
2302        /*: assume "0 <= i2 & i2 < sb..csize" */
2303        Object r2b = sb.set(i2, v2);
2304        int r1b = sb.indexOf(v1);
2305
2306        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
2307    }
2308
2309    static void indexOf_set_pre_s_75(ArrayList sa, ArrayList sb, Object v1, int i2,
            Object v2)
2310    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2311                  sa..contents = sb..contents & sa..csize = sb..csize"
2312        modifies "sa..contents", "sb..contents"
2313        ensures "True" */
2314    {
2315        /*: assume "(~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
                v2) | (EX i. (i, v1) : sa..contents & 0 <= i & i < i2) | (~(EX i. (i, v1) :
                sa..contents & 0 <= i & i < i2) & (i2, v1) : sa..contents & v1 = v2) | (~(EX
                i. (i, v1) : sa..contents & 0 <= i & i <= i2) & (EX i. (i, v1) :
                sa..contents & i2 < i & i < sa..csize) & v1 ~= v2)" */
2316        int r1a = sa.indexOf(v1);
2317        /*: assume "0 <= i2 & i2 < sa..csize" */
2318        sa.set(i2, v2);
2319
2320        sb.set(i2, v2);
2321        int r1b = sb.indexOf(v1);
2322
2323        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
2324    }
2325
2326    static void indexOf_set_pre_c_75(ArrayList sa, ArrayList sb, Object v1, int i2,
            Object v2)
2327    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2328                  sa..contents = sb..contents & sa..csize = sb..csize"
2329        modifies "sa..contents", "sb..contents"
2330        ensures "True" */
2331    {
2332        /*: assume "~((~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
                v2) | (EX i. (i, v1) : sa..contents & 0 <= i & i < i2) | (~(EX i. (i, v1) :
                sa..contents & 0 <= i & i < i2) & (i2, v1) : sa..contents & v1 = v2) | (~(EX
                i. (i, v1) : sa..contents & 0 <= i & i <= i2) & (EX i. (i, v1) :
                sa..contents & i2 < i & i < sa..csize) & v1 ~= v2))" */
2333        int r1a = sa.indexOf(v1);
2334        /*: assume "0 <= i2 & i2 < sa..csize" */
2335        sa.set(i2, v2);
2336
2337        /*: assume "0 <= i2 & i2 < sb..csize" */
2338        sb.set(i2, v2);
2339        int r1b = sb.indexOf(v1);
2340
2341        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
2342    }
2343
2344    static void indexOf_set_between_s_76(ArrayList sa, ArrayList sb, Object v1, int i2,
            Object v2)
```

```
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 = v2)
            | (r1a > i2 & v1 ~= v2)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        sb.set(i2, v2);
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void indexOf_set_between_c_76(ArrayList sa, ArrayList sb, Object v1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 =
            v2) | (r1a > i2 & v1 ~= v2))" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.set(i2, v2);
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void indexOf_set_post_s_77(ArrayList sa, ArrayList sb, Object v1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);
        /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 = v2)
            | (r1a > i2 & v1 ~= v2)" */

        sb.set(i2, v2);
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void indexOf_set_post_c_77(ArrayList sa, ArrayList sb, Object v1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
```

```
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);
        /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 =
            v2) | (r1a > i2 & v1 ~= v2))" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.set(i2, v2);
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void indexOf_size_pre_s_78(ArrayList sa, ArrayList sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "True" */
        int r1a = sa.indexOf(v1);
        int r2a = sa.size();

        int r2b = sb.size();
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void indexOf_size_pre_c_78(ArrayList sa, ArrayList sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "~(True)" */
        int r1a = sa.indexOf(v1);
        int r2a = sa.size();

        int r2b = sb.size();
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void indexOf_size_between_s_79(ArrayList sa, ArrayList sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        /*: assume "True" */
        int r2a = sa.size();

        int r2b = sb.size();
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void indexOf_size_between_c_79(ArrayList sa, ArrayList sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
```

```
    {
        int r1a = sa.indexOf(v1);
        /*: assume "~(True)" */
        int r2a = sa.size();

        int r2b = sb.size();
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void indexOf_size_post_s_80(ArrayList sa, ArrayList sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        int r2a = sa.size();
        /*: assume "True" */

        int r2b = sb.size();
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void indexOf_size_post_c_80(ArrayList sa, ArrayList sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.indexOf(v1);
        int r2a = sa.size();
        /*: assume "~(True)" */

        int r2b = sb.size();
        int r1b = sb.indexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void lastIndexOf_add_at_pre_s_81(ArrayList sa, ArrayList sb, Object v1, int
        i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "(~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
            v2) | ((EX i. (i, v1) : sa..contents & 0 <= i & i < i2) & ~(EX i. (i, v1) :
            sa..contents & i2 <= i & i < sa..csize) & v1 ~= v2)" */
        int r1a = sa.lastIndexOf(v1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        sb.add_at(i2, v2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }
```

```
    static void lastIndexOf_add_at_between_s_82(ArrayList sa, ArrayList sb, Object v1,
        int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        int r1a = sa.lastIndexOf(v1);
        /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2)" */
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        sb.add_at(i2, v2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void lastIndexOf_add_at_post_s_83(ArrayList sa, ArrayList sb, Object v1, int
        i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        int r1a = sa.lastIndexOf(v1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);
        /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2)" */

        sb.add_at(i2, v2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void lastIndexOf_get_pre_s_84(ArrayList sa, ArrayList sb, Object v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "True" */
        int r1a = sa.lastIndexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        Object r2b = sb.get(i2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void lastIndexOf_get_pre_c_84(ArrayList sa, ArrayList sb, Object v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "~(True)" */
        int r1a = sa.lastIndexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

```

```
        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.get(i2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void lastIndexOf_get_between_s_85(ArrayList sa, ArrayList sb, Object v1, int
        i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.lastIndexOf(v1);
        /*: assume "True" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        Object r2b = sb.get(i2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void lastIndexOf_get_between_c_85(ArrayList sa, ArrayList sb, Object v1, int
        i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.lastIndexOf(v1);
        /*: assume "~(True)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.get(i2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void lastIndexOf_get_post_s_86(ArrayList sa, ArrayList sb, Object v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.lastIndexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);
        /*: assume "True" */

        Object r2b = sb.get(i2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void lastIndexOf_get_post_c_86(ArrayList sa, ArrayList sb, Object v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
```

```
        ensures "True" */
    {
        int r1a = sa.lastIndexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);
        /*: assume "~(True)" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.get(i2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void lastIndexOf_indexOf_pre_s_87(ArrayList sa, ArrayList sb, Object v1,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "True" */
        int r1a = sa.lastIndexOf(v1);
        int r2a = sa.indexOf(v2);

        int r2b = sb.indexOf(v2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void lastIndexOf_indexOf_pre_c_87(ArrayList sa, ArrayList sb, Object v1,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "~(True)" */
        int r1a = sa.lastIndexOf(v1);
        int r2a = sa.indexOf(v2);

        int r2b = sb.indexOf(v2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void lastIndexOf_indexOf_between_s_88(ArrayList sa, ArrayList sb, Object v1,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.lastIndexOf(v1);
        /*: assume "True" */
        int r2a = sa.indexOf(v2);

        int r2b = sb.indexOf(v2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }
```

```
2698
2699    static void lastIndexOf_indexOf_between_c_88(ArrayList sa, ArrayList sb, Object v1,
            Object v2)
2700    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2701                sa..contents = sb..contents & sa..csize = sb..csize"
2702        ensures "True" */
2703    {
2704        int r1a = sa.lastIndexOf(v1);
2705        /*: assume "~(True)" */
2706        int r2a = sa.indexOf(v2);
2707
2708        int r2b = sb.indexOf(v2);
2709        int r1b = sb.lastIndexOf(v1);
2710
2711        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
2712    }
2713
2714    static void lastIndexOf_indexOf_post_s_89(ArrayList sa, ArrayList sb, Object v1,
            Object v2)
2715    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2716                sa..contents = sb..contents & sa..csize = sb..csize"
2717        ensures "True" */
2718    {
2719        int r1a = sa.lastIndexOf(v1);
2720        int r2a = sa.indexOf(v2);
2721        /*: assume "True" */
2722
2723        int r2b = sb.indexOf(v2);
2724        int r1b = sb.lastIndexOf(v1);
2725
2726        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
2727    }
2728
2729    static void lastIndexOf_indexOf_post_c_89(ArrayList sa, ArrayList sb, Object v1,
            Object v2)
2730    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2731                sa..contents = sb..contents & sa..csize = sb..csize"
2732        ensures "True" */
2733    {
2734        int r1a = sa.lastIndexOf(v1);
2735        int r2a = sa.indexOf(v2);
2736        /*: assume "~(True)" */
2737
2738        int r2b = sb.indexOf(v2);
2739        int r1b = sb.lastIndexOf(v1);
2740
2741        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
2742    }
2743
2744    static void lastIndexOf_lastIndexOf_pre_s_90(ArrayList sa, ArrayList sb, Object v1,
            Object v2)
2745    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2746                sa..contents = sb..contents & sa..csize = sb..csize"
2747        ensures "True" */
2748    {
2749        /*: assume "True" */
2750        int r1a = sa.lastIndexOf(v1);
2751        int r2a = sa.lastIndexOf(v2);
2752
2753        int r2b = sb.lastIndexOf(v2);
2754        int r1b = sb.lastIndexOf(v1);
2755
```

```
2756        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
2757    }
2758
2759    static void lastIndexOf_lastIndexOf_pre_c_90(ArrayList sa, ArrayList sb, Object v1,
            Object v2)
2760    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2761                  sa..contents = sb..contents & sa..csize = sb..csize"
2762        ensures "True" */
2763    {
2764        /*: assume "~(True)" */
2765        int r1a = sa.lastIndexOf(v1);
2766        int r2a = sa.lastIndexOf(v2);
2767
2768        int r2b = sb.lastIndexOf(v2);
2769        int r1b = sb.lastIndexOf(v1);
2770
2771        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
2772    }
2773
2774    static void lastIndexOf_lastIndexOf_between_s_91(ArrayList sa, ArrayList sb, Object
            v1, Object v2)
2775    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2776                  sa..contents = sb..contents & sa..csize = sb..csize"
2777        ensures "True" */
2778    {
2779        int r1a = sa.lastIndexOf(v1);
2780        /*: assume "True" */
2781        int r2a = sa.lastIndexOf(v2);
2782
2783        int r2b = sb.lastIndexOf(v2);
2784        int r1b = sb.lastIndexOf(v1);
2785
2786        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
2787    }
2788
2789    static void lastIndexOf_lastIndexOf_between_c_91(ArrayList sa, ArrayList sb, Object
            v1, Object v2)
2790    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2791                  sa..contents = sb..contents & sa..csize = sb..csize"
2792        ensures "True" */
2793    {
2794        int r1a = sa.lastIndexOf(v1);
2795        /*: assume "~(True)" */
2796        int r2a = sa.lastIndexOf(v2);
2797
2798        int r2b = sb.lastIndexOf(v2);
2799        int r1b = sb.lastIndexOf(v1);
2800
2801        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
2802    }
2803
2804    static void lastIndexOf_lastIndexOf_post_s_92(ArrayList sa, ArrayList sb, Object
            v1, Object v2)
2805    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2806                  sa..contents = sb..contents & sa..csize = sb..csize"
2807        ensures "True" */
2808    {
2809        int r1a = sa.lastIndexOf(v1);
2810        int r2a = sa.lastIndexOf(v2);
2811        /*: assume "True" */
2812
```

```
2813        int r2b = sb.lastIndexOf(v2);
2814        int r1b = sb.lastIndexOf(v1);
2815
2816        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
2817    }
2818
2819    static void lastIndexOf_lastIndexOf_post_c_92(ArrayList sa, ArrayList sb, Object
            v1, Object v2)
2820    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2821                sa..contents = sb..contents & sa..csize = sb..csize"
2822        ensures "True" */
2823    {
2824        int r1a = sa.lastIndexOf(v1);
2825        int r2a = sa.lastIndexOf(v2);
2826        /*: assume "~(True)" */
2827
2828        int r2b = sb.lastIndexOf(v2);
2829        int r1b = sb.lastIndexOf(v1);
2830
2831        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
2832    }
2833
2834    static void lastIndexOf_remove_at_pre_s_93(ArrayList sa, ArrayList sb, Object v1,
            int i2)
2835    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2836                sa..contents = sb..contents & sa..csize = sb..csize"
2837        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2838        ensures "True" */
2839    {
2840        /*: assume "~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) | ((EX i.
                (i, v1) : sa..contents & 0 <= i & i < i2) & ~(EX i. (i, v1) : sa..contents &
                i2 <= i & i < sa..csize))" */
2841        int r1a = sa.lastIndexOf(v1);
2842        /*: assume "0 <= i2 & i2 < sa..csize" */
2843        Object r2a = sa.remove_at(i2);
2844
2845        Object r2b = sb.remove_at(i2);
2846        int r1b = sb.lastIndexOf(v1);
2847
2848        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
2849    }
2850
2851    static void lastIndexOf_remove_at_pre_c_93(ArrayList sa, ArrayList sb, Object v1,
            int i2)
2852    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2853                sa..contents = sb..contents & sa..csize = sb..csize"
2854        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2855        ensures "True" */
2856    {
2857        /*: assume "~(~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) | ((EX
                i. (i, v1) : sa..contents & 0 <= i & i < i2) & ~(EX i. (i, v1) :
                sa..contents & i2 <= i & i < sa..csize)))" */
2858        int r1a = sa.lastIndexOf(v1);
2859        /*: assume "0 <= i2 & i2 < sa..csize" */
2860        Object r2a = sa.remove_at(i2);
2861
2862        /*: assume "0 <= i2 & i2 < sb..csize" */
2863        Object r2b = sb.remove_at(i2);
2864        int r1b = sb.lastIndexOf(v1);
2865
2866        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
```

```
2867    }
2868
2869    static void lastIndexOf_remove_at_between_s_94(ArrayList sa, ArrayList sb, Object
            v1, int i2)
2870    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2871                sa..contents = sb..contents & sa..csize = sb..csize"
2872        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2873        ensures "True" */
2874    {
2875        int r1a = sa.lastIndexOf(v1);
2876        /*: assume "r1a < 0 | (0 <= r1a & r1a < i2)" */
2877        /*: assume "0 <= i2 & i2 < sa..csize" */
2878        Object r2a = sa.remove_at(i2);
2879
2880        Object r2b = sb.remove_at(i2);
2881        int r1b = sb.lastIndexOf(v1);
2882
2883        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
2884    }
2885
2886    static void lastIndexOf_remove_at_between_c_94(ArrayList sa, ArrayList sb, Object
            v1, int i2)
2887    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2888                sa..contents = sb..contents & sa..csize = sb..csize"
2889        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2890        ensures "True" */
2891    {
2892        int r1a = sa.lastIndexOf(v1);
2893        /*: assume "~(r1a < 0 | (0 <= r1a & r1a < i2))" */
2894        /*: assume "0 <= i2 & i2 < sa..csize" */
2895        Object r2a = sa.remove_at(i2);
2896
2897        /*: assume "0 <= i2 & i2 < sb..csize" */
2898        Object r2b = sb.remove_at(i2);
2899        int r1b = sb.lastIndexOf(v1);
2900
2901        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
2902    }
2903
2904    static void lastIndexOf_remove_at_post_s_95(ArrayList sa, ArrayList sb, Object v1,
            int i2)
2905    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2906                sa..contents = sb..contents & sa..csize = sb..csize"
2907        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2908        ensures "True" */
2909    {
2910        int r1a = sa.lastIndexOf(v1);
2911        /*: assume "0 <= i2 & i2 < sa..csize" */
2912        Object r2a = sa.remove_at(i2);
2913        /*: assume "r1a < 0 | (0 <= r1a & r1a < i2)" */
2914
2915        Object r2b = sb.remove_at(i2);
2916        int r1b = sb.lastIndexOf(v1);
2917
2918        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
2919    }
2920
2921    static void lastIndexOf_remove_at_post_c_95(ArrayList sa, ArrayList sb, Object v1,
            int i2)
2922    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2923                sa..contents = sb..contents & sa..csize = sb..csize"
2924        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
```

```
2925        ensures "True" */
2926    {
2927        int r1a = sa.lastIndexOf(v1);
2928        /*: assume "0 <= i2 & i2 < sa..csize" */
2929        Object r2a = sa.remove_at(i2);
2930        /*: assume "~(r1a < 0 | (0 <= r1a & r1a < i2))" */
2931
2932        /*: assume "0 <= i2 & i2 < sb..csize" */
2933        Object r2b = sb.remove_at(i2);
2934        int r1b = sb.lastIndexOf(v1);
2935
2936        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
2937    }
2938
2939    static void lastIndexOf_remove_at_pre_s_96(ArrayList sa, ArrayList sb, Object v1,
            int i2)
2940    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2941                  sa..contents = sb..contents & sa..csize = sb..csize"
2942        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2943        ensures "True" */
2944    {
2945        /*: assume "~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) | ((EX i.
                (i, v1) : sa..contents & 0 <= i & i < i2) & ~(EX i. (i, v1) : sa..contents &
                i2 <= i & i < sa..csize))" */
2946        int r1a = sa.lastIndexOf(v1);
2947        /*: assume "0 <= i2 & i2 < sa..csize" */
2948        sa.remove_at(i2);
2949
2950        sb.remove_at(i2);
2951        int r1b = sb.lastIndexOf(v1);
2952
2953        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
2954    }
2955
2956    static void lastIndexOf_remove_at_pre_c_96(ArrayList sa, ArrayList sb, Object v1,
            int i2)
2957    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2958                  sa..contents = sb..contents & sa..csize = sb..csize"
2959        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2960        ensures "True" */
2961    {
2962        /*: assume "~(~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) | ((EX
                i. (i, v1) : sa..contents & 0 <= i & i < i2) & ~(EX i. (i, v1) :
                sa..contents & i2 <= i & i < sa..csize)))" */
2963        int r1a = sa.lastIndexOf(v1);
2964        /*: assume "0 <= i2 & i2 < sa..csize" */
2965        sa.remove_at(i2);
2966
2967        /*: assume "0 <= i2 & i2 < sb..csize" */
2968        sb.remove_at(i2);
2969        int r1b = sb.lastIndexOf(v1);
2970
2971        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
2972    }
2973
2974    static void lastIndexOf_remove_at_between_s_97(ArrayList sa, ArrayList sb, Object
            v1, int i2)
2975    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2976                  sa..contents = sb..contents & sa..csize = sb..csize"
2977        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2978        ensures "True" */
2979    {
2980        int r1a = sa.lastIndexOf(v1);
```

```
        /*: assume "r1a < 0 | (0 <= r1a & r1a < i2)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        sb.remove_at(i2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void lastIndexOf_remove_at_between_c_97(ArrayList sa, ArrayList sb, Object
        v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        int r1a = sa.lastIndexOf(v1);
        /*: assume "~(r1a < 0 | (0 <= r1a & r1a < i2))" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void lastIndexOf_remove_at_post_s_98(ArrayList sa, ArrayList sb, Object v1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        int r1a = sa.lastIndexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);
        /*: assume "r1a < 0 | (0 <= r1a & r1a < i2)" */

        sb.remove_at(i2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void lastIndexOf_remove_at_post_c_98(ArrayList sa, ArrayList sb, Object v1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        int r1a = sa.lastIndexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);
        /*: assume "~(r1a < 0 | (0 <= r1a & r1a < i2))" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        int r1b = sb.lastIndexOf(v1);

```

```
        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void lastIndexOf_set_pre_s_99(ArrayList sa, ArrayList sb, Object v1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "(~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
            v2) | ((EX i. (i, v1) : sa..contents & 0 <= i & i < i2) & ~(EX i. (i, v1) :
            sa..contents & i2 <= i & i < sa..csize) & v1 ~= v2) | ((i2, v1) :
            sa..contents & 0 <= i2 & i2 < sa..csize & ~(EX i. (i, v1) : sa..contents &
            i2 < i & i < sa..csize) & v1 = v2) | (EX i. (i, v1) : sa..contents & i2 < i
            & i < sa..csize)" */
        int r1a = sa.lastIndexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        Object r2b = sb.set(i2, v2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void lastIndexOf_set_pre_c_99(ArrayList sa, ArrayList sb, Object v1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "~((~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
            v2) | ((EX i. (i, v1) : sa..contents & 0 <= i & i < i2) & ~(EX i. (i, v1) :
            sa..contents & i2 <= i & i < sa..csize) & v1 ~= v2) | ((i2, v1) :
            sa..contents & 0 <= i2 & i2 < sa..csize & ~(EX i. (i, v1) : sa..contents &
            i2 < i & i < sa..csize) & v1 = v2) | (EX i. (i, v1) : sa..contents & i2 < i
            & i < sa..csize))" */
        int r1a = sa.lastIndexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.set(i2, v2);
        int r1b = sb.lastIndexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void lastIndexOf_set_between_s_100(ArrayList sa, ArrayList sb, Object v1,
        int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        int r1a = sa.lastIndexOf(v1);
        /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2) | (r1a = i2
            & v1 = v2) | r1a > i2" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);
```

```
3089
3090            Object r2b = sb.set(i2, v2);
3091            int r1b = sb.lastIndexOf(v1);
3092
3093            /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                    sb..csize" */
3094        }
3095
3096        static void lastIndexOf_set_between_c_100(ArrayList sa, ArrayList sb, Object v1,
                int i2, Object v2)
3097        /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3098                      sa..contents = sb..contents & sa..csize = sb..csize"
3099            modifies "sa..contents", "sb..contents"
3100            ensures "True" */
3101        {
3102            int r1a = sa.lastIndexOf(v1);
3103            /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2) | (r1a =
                    i2 & v1 = v2) | r1a > i2)" */
3104            /*: assume "0 <= i2 & i2 < sa..csize" */
3105            Object r2a = sa.set(i2, v2);
3106
3107            /*: assume "0 <= i2 & i2 < sb..csize" */
3108            Object r2b = sb.set(i2, v2);
3109            int r1b = sb.lastIndexOf(v1);
3110
3111            /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                    sb..csize)" */
3112        }
3113
3114        static void lastIndexOf_set_post_s_101(ArrayList sa, ArrayList sb, Object v1, int
                i2, Object v2)
3115        /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3116                      sa..contents = sb..contents & sa..csize = sb..csize"
3117            modifies "sa..contents", "sb..contents"
3118            ensures "True" */
3119        {
3120            int r1a = sa.lastIndexOf(v1);
3121            /*: assume "0 <= i2 & i2 < sa..csize" */
3122            Object r2a = sa.set(i2, v2);
3123            /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2) | (r1a = i2
                    & v1 = v2) | r1a > i2" */
3124
3125            Object r2b = sb.set(i2, v2);
3126            int r1b = sb.lastIndexOf(v1);
3127
3128            /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                    sb..csize" */
3129        }
3130
3131        static void lastIndexOf_set_post_c_101(ArrayList sa, ArrayList sb, Object v1, int
                i2, Object v2)
3132        /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3133                      sa..contents = sb..contents & sa..csize = sb..csize"
3134            modifies "sa..contents", "sb..contents"
3135            ensures "True" */
3136        {
3137            int r1a = sa.lastIndexOf(v1);
3138            /*: assume "0 <= i2 & i2 < sa..csize" */
3139            Object r2a = sa.set(i2, v2);
3140            /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2) | (r1a =
                    i2 & v1 = v2) | r1a > i2)" */
3141
3142            /*: assume "0 <= i2 & i2 < sb..csize" */
3143            Object r2b = sb.set(i2, v2);
3144            int r1b = sb.lastIndexOf(v1);
```

```
3145
3146        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
3147    }
3148
3149    static void lastIndexOf_set_pre_s_102(ArrayList sa, ArrayList sb, Object v1, int
            i2, Object v2)
3150    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3151                  sa..contents = sb..contents & sa..csize = sb..csize"
3152        modifies "sa..contents", "sb..contents"
3153        ensures "True" */
3154    {
3155        /*: assume "(~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
                v2) | ((EX i. (i, v1) : sa..contents & 0 <= i & i < i2) & ~(EX i. (i, v1) :
                sa..contents & i2 <= i & i < sa..csize) & v1 ~= v2) | ((i2, v1) :
                sa..contents & 0 <= i2 & i2 < sa..csize & ~(EX i. (i, v1) : sa..contents &
                i2 < i & i < sa..csize) & v1 = v2) | (EX i. (i, v1) : sa..contents & i2 < i
                & i < sa..csize)" */
3156        int r1a = sa.lastIndexOf(v1);
3157        /*: assume "0 <= i2 & i2 < sa..csize" */
3158        sa.set(i2, v2);
3159
3160        sb.set(i2, v2);
3161        int r1b = sb.lastIndexOf(v1);
3162
3163        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
3164    }
3165
3166    static void lastIndexOf_set_pre_c_102(ArrayList sa, ArrayList sb, Object v1, int
            i2, Object v2)
3167    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3168                  sa..contents = sb..contents & sa..csize = sb..csize"
3169        modifies "sa..contents", "sb..contents"
3170        ensures "True" */
3171    {
3172        /*: assume "~((~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
                v2) | ((EX i. (i, v1) : sa..contents & 0 <= i & i < i2) & ~(EX i. (i, v1) :
                sa..contents & i2 <= i & i < sa..csize) & v1 ~= v2) | ((i2, v1) :
                sa..contents & 0 <= i2 & i2 < sa..csize & ~(EX i. (i, v1) : sa..contents &
                i2 < i & i < sa..csize) & v1 = v2) | (EX i. (i, v1) : sa..contents & i2 < i
                & i < sa..csize))" */
3173        int r1a = sa.lastIndexOf(v1);
3174        /*: assume "0 <= i2 & i2 < sa..csize" */
3175        sa.set(i2, v2);
3176
3177        /*: assume "0 <= i2 & i2 < sb..csize" */
3178        sb.set(i2, v2);
3179        int r1b = sb.lastIndexOf(v1);
3180
3181        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
3182    }
3183
3184    static void lastIndexOf_set_between_s_103(ArrayList sa, ArrayList sb, Object v1,
            int i2, Object v2)
3185    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3186                  sa..contents = sb..contents & sa..csize = sb..csize"
3187        modifies "sa..contents", "sb..contents"
3188        ensures "True" */
3189    {
3190        int r1a = sa.lastIndexOf(v1);
3191        /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2) | (r1a = i2
                & v1 = v2) | r1a > i2" */
3192        /*: assume "0 <= i2 & i2 < sa..csize" */
3193        sa.set(i2, v2);
```

```
3194
3195        sb.set(i2, v2);
3196        int r1b = sb.lastIndexOf(v1);
3197
3198        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
3199    }
3200
3201    static void lastIndexOf_set_between_c_103(ArrayList sa, ArrayList sb, Object v1,
        int i2, Object v2)
3202    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3203                sa..contents = sb..contents & sa..csize = sb..csize"
3204        modifies "sa..contents", "sb..contents"
3205        ensures "True" */
3206    {
3207        int r1a = sa.lastIndexOf(v1);
3208        /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2) | (r1a =
            i2 & v1 = v2) | r1a > i2)" */
3209        /*: assume "0 <= i2 & i2 < sa..csize" */
3210        sa.set(i2, v2);
3211
3212        /*: assume "0 <= i2 & i2 < sb..csize" */
3213        sb.set(i2, v2);
3214        int r1b = sb.lastIndexOf(v1);
3215
3216        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
3217    }
3218
3219    static void lastIndexOf_set_post_s_104(ArrayList sa, ArrayList sb, Object v1, int
        i2, Object v2)
3220    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3221                sa..contents = sb..contents & sa..csize = sb..csize"
3222        modifies "sa..contents", "sb..contents"
3223        ensures "True" */
3224    {
3225        int r1a = sa.lastIndexOf(v1);
3226        /*: assume "0 <= i2 & i2 < sa..csize" */
3227        sa.set(i2, v2);
3228        /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2) | (r1a = i2
            & v1 = v2) | r1a > i2" */
3229
3230        sb.set(i2, v2);
3231        int r1b = sb.lastIndexOf(v1);
3232
3233        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
3234    }
3235
3236    static void lastIndexOf_set_post_c_104(ArrayList sa, ArrayList sb, Object v1, int
        i2, Object v2)
3237    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3238                sa..contents = sb..contents & sa..csize = sb..csize"
3239        modifies "sa..contents", "sb..contents"
3240        ensures "True" */
3241    {
3242        int r1a = sa.lastIndexOf(v1);
3243        /*: assume "0 <= i2 & i2 < sa..csize" */
3244        sa.set(i2, v2);
3245        /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2) | (r1a =
            i2 & v1 = v2) | r1a > i2)" */
3246
3247        /*: assume "0 <= i2 & i2 < sb..csize" */
3248        sb.set(i2, v2);
3249        int r1b = sb.lastIndexOf(v1);
3250
```

```
        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void lastIndexOf_size_pre_s_105(ArrayList sa, ArrayList sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "True" */
        int r1a = sa.lastIndexOf(v1);
        int r2a = sa.size();

        int r2b = sb.size();
        int r1b = sb.lastIndexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void lastIndexOf_size_pre_c_105(ArrayList sa, ArrayList sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "~(True)" */
        int r1a = sa.lastIndexOf(v1);
        int r2a = sa.size();

        int r2b = sb.size();
        int r1b = sb.lastIndexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void lastIndexOf_size_between_s_106(ArrayList sa, ArrayList sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.lastIndexOf(v1);
        /*: assume "True" */
        int r2a = sa.size();

        int r2b = sb.size();
        int r1b = sb.lastIndexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void lastIndexOf_size_between_c_106(ArrayList sa, ArrayList sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.lastIndexOf(v1);
        /*: assume "~(True)" */
        int r2a = sa.size();

        int r2b = sb.size();
        int r1b = sb.lastIndexOf(v1);

```

```
        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void lastIndexOf_size_post_s_107(ArrayList sa, ArrayList sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.lastIndexOf(v1);
        int r2a = sa.size();
        /*: assume "True" */

        int r2b = sb.size();
        int r1b = sb.lastIndexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void lastIndexOf_size_post_c_107(ArrayList sa, ArrayList sb, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.lastIndexOf(v1);
        int r2a = sa.size();
        /*: assume "~(True)" */

        int r2b = sb.size();
        int r1b = sb.lastIndexOf(v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void remove_at_add_at_pre_s_108(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "(i1 < i2 & (i2, v2) : sa..contents & 0 <= i2 & i2 < sa..csize) |
            (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize) | (i1 > i2 &
            (ALL v. ((i1 - 1, v) : sa..contents) = ((i1, v) : sa..contents)) & 0 <= i1 -
            1 & i1 - 1 < sa..csize & 0 <= i1 & i1 < sa..csize)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        sb.add_at(i2, v2);
        Object r1b = sb.remove_at(i1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_add_at_pre_c_108(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
```

```
3366        ensures "True" */
3367    {
3368        /*: assume "~((i1 < i2 & (i2, v2) : sa..contents & 0 <= i2 & i2 < sa..csize) |
                (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize) | (i1 > i2 &
                (ALL v. ((i1 - 1, v) : sa..contents) = ((i1, v) : sa..contents)) & 0 <= i1 -
                1 & i1 - 1 < sa..csize & 0 <= i1 & i1 < sa..csize))" */
3369        /*: assume "0 <= i1 & i1 < sa..csize" */
3370        Object r1a = sa.remove_at(i1);
3371        /*: assume "0 <= i2 & i2 <= sa..csize" */
3372        sa.add_at(i2, v2);
3373
3374        /*: assume "0 <= i2 & i2 <= sb..csize" */
3375        sb.add_at(i2, v2);
3376        /*: assume "0 <= i1 & i1 < sb..csize" */
3377        Object r1b = sb.remove_at(i1);
3378
3379        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
3380    }
3381
3382    static void remove_at_add_at_between_s_109(ArrayList sa, ArrayList sb, int i1, int
        i2, Object v2)
3383    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3384                  sa..contents = sb..contents & sa..csize = sb..csize"
3385        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
3386        ensures "True" */
3387    {
3388        /*: assume "0 <= i1 & i1 < sa..csize" */
3389        Object r1a = sa.remove_at(i1);
3390        /*: assume "(i1 < i2 & (i2 - 1, v2) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
            sa..csize) | (i1 = i2 & r1a = v2) | (i1 > i2 & (i1 - 1, r1a) : sa..contents
            & 0 <= i1 - 1 & i1 - 1 < sa..csize)" */
3391        /*: assume "0 <= i2 & i2 <= sa..csize" */
3392        sa.add_at(i2, v2);
3393
3394        sb.add_at(i2, v2);
3395        Object r1b = sb.remove_at(i1);
3396
3397        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
3398    }
3399
3400    static void remove_at_add_at_between_c_109(ArrayList sa, ArrayList sb, int i1, int
        i2, Object v2)
3401    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3402                  sa..contents = sb..contents & sa..csize = sb..csize"
3403        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
3404        ensures "True" */
3405    {
3406        /*: assume "0 <= i1 & i1 < sa..csize" */
3407        Object r1a = sa.remove_at(i1);
3408        /*: assume "~((i1 < i2 & (i2 - 1, v2) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
            sa..csize) | (i1 = i2 & r1a = v2) | (i1 > i2 & (i1 - 1, r1a) : sa..contents
            & 0 <= i1 - 1 & i1 - 1 < sa..csize))" */
3409        /*: assume "0 <= i2 & i2 <= sa..csize" */
3410        sa.add_at(i2, v2);
3411
3412        /*: assume "0 <= i2 & i2 <= sb..csize" */
3413        sb.add_at(i2, v2);
3414        /*: assume "0 <= i1 & i1 < sb..csize" */
3415        Object r1b = sb.remove_at(i1);
3416
3417        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
```

```
3418    }
3419
3420    static void remove_at_add_at_post_s_110(ArrayList sa, ArrayList sb, int i1, int i2,
            Object v2)
3421    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3422                sa..contents = sb..contents & sa..csize = sb..csize"
3423        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
3424        ensures "True" */
3425    {
3426        /*: assume "0 <= i1 & i1 < sa..csize" */
3427        Object r1a = sa.remove_at(i1);
3428        /*: assume "0 <= i2 & i2 <= sa..csize" */
3429        sa.add_at(i2, v2);
3430        /*: assume "(i1 < i2 & (i2 - 1, v2) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
                sa..csize) | (i1 = i2 & r1a = v2) | (i1 > i2 & (i1, r1a) : sa..contents & 0
                <= i1 & i1 < sa..csize)" */
3431
3432        sb.add_at(i2, v2);
3433        Object r1b = sb.remove_at(i1);
3434
3435        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
3436    }
3437
3438    static void remove_at_add_at_post_c_110(ArrayList sa, ArrayList sb, int i1, int i2,
            Object v2)
3439    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3440                sa..contents = sb..contents & sa..csize = sb..csize"
3441        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
3442        ensures "True" */
3443    {
3444        /*: assume "0 <= i1 & i1 < sa..csize" */
3445        Object r1a = sa.remove_at(i1);
3446        /*: assume "0 <= i2 & i2 <= sa..csize" */
3447        sa.add_at(i2, v2);
3448        /*: assume "~((i1 < i2 & (i2 - 1, v2) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
                sa..csize) | (i1 = i2 & r1a = v2) | (i1 > i2 & (i1, r1a) : sa..contents & 0
                <= i1 & i1 < sa..csize))" */
3449
3450        /*: assume "0 <= i2 & i2 <= sb..csize" */
3451        sb.add_at(i2, v2);
3452        /*: assume "0 <= i1 & i1 < sb..csize" */
3453        Object r1b = sb.remove_at(i1);
3454
3455        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
3456    }
3457
3458    static void remove_at_get_pre_s_111(ArrayList sa, ArrayList sb, int i1, int i2)
3459    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3460                sa..contents = sb..contents & sa..csize = sb..csize"
3461        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3462        ensures "True" */
3463    {
3464        /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
                sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
                sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :
                sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
                sa..csize) | i1 > i2" */
3465        /*: assume "0 <= i1 & i1 < sa..csize" */
3466        Object r1a = sa.remove_at(i1);
3467        /*: assume "0 <= i2 & i2 < sa..csize" */
3468        Object r2a = sa.get(i2);
3469
```

```
        Object r2b = sb.get(i2);
        Object r1b = sb.remove_at(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void remove_at_get_pre_c_111(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | i1 > i2)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.get(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.remove_at(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void remove_at_get_between_s_112(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
            sa..csize) | (i1 = i2 & (i2, r1a) : sa..contents) | i1 > i2" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        Object r2b = sb.get(i2);
        Object r1b = sb.remove_at(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void remove_at_get_between_c_112(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
            sa..csize) | (i1 = i2 & (i2, r1a) : sa..contents) | i1 > i2)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
```

```
        Object r2a = sa.get(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.get(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.remove_at(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void remove_at_get_post_s_113(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);
        /*: assume "(i1 < i2 & (i2 - 1, r2a) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
            sa..csize) | (i1 = i2 & r1a = r2a) | i1 > i2" */

        Object r2b = sb.get(i2);
        Object r1b = sb.remove_at(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void remove_at_get_post_c_113(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);
        /*: assume "~((i1 < i2 & (i2 - 1, r2a) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
            sa..csize) | (i1 = i2 & r1a = r2a) | i1 > i2)" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.get(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.remove_at(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void remove_at_lastIndexOf_pre_s_117(ArrayList sa, ArrayList sb, int i1,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | ((EX i.
            (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) : sa..contents &
            i1 <= i & i < sa..csize))" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
```

```
3581        int r2a = sa.lastIndexOf(v2);
3582
3583        int r2b = sb.lastIndexOf(v2);
3584        Object r1b = sb.remove_at(i1);
3585
3586        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
3587    }
3588
3589    static void remove_at_lastIndexOf_pre_c_117(ArrayList sa, ArrayList sb, int i1,
            Object v2)
3590    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3591                  sa..contents = sb..contents & sa..csize = sb..csize"
3592        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3593        ensures "True" */
3594    {
3595        /*: assume "~(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | ((EX
                i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
                sa..contents & i1 <= i & i < sa..csize)))" */
3596        /*: assume "0 <= i1 & i1 < sa..csize" */
3597        Object r1a = sa.remove_at(i1);
3598        int r2a = sa.lastIndexOf(v2);
3599
3600        int r2b = sb.lastIndexOf(v2);
3601        /*: assume "0 <= i1 & i1 < sb..csize" */
3602        Object r1b = sb.remove_at(i1);
3603
3604        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
3605    }
3606
3607    static void remove_at_lastIndexOf_between_c_118(ArrayList sa, ArrayList sb, int i1,
            Object v2)
3608    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3609                  sa..contents = sb..contents & sa..csize = sb..csize"
3610        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3611        ensures "True" */
3612    {
3613        /*: assume "0 <= i1 & i1 < sa..csize" */
3614        Object r1a = sa.remove_at(i1);
3615        /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & r1a
                ~= v2) | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2)
                : sa..contents & i1 <= i & i < sa..csize) & r1a ~= v2))" */
3616        int r2a = sa.lastIndexOf(v2);
3617
3618        int r2b = sb.lastIndexOf(v2);
3619        /*: assume "0 <= i1 & i1 < sb..csize" */
3620        Object r1b = sb.remove_at(i1);
3621
3622        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
3623    }
3624
3625    static void remove_at_lastIndexOf_post_c_119(ArrayList sa, ArrayList sb, int i1,
            Object v2)
3626    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3627                  sa..contents = sb..contents & sa..csize = sb..csize"
3628        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3629        ensures "True" */
3630    {
3631        /*: assume "0 <= i1 & i1 < sa..csize" */
3632        Object r1a = sa.remove_at(i1);
3633        int r2a = sa.lastIndexOf(v2);
3634        /*: assume "~((r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1 & r1a ~= v2))" */
3635
```

```
3636        int r2b = sb.lastIndexOf(v2);
3637        /*: assume "0 <= i1 & i1 < sb..csize" */
3638        Object r1b = sb.remove_at(i1);
3639
3640        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
3641    }
3642
3643    static void remove_at_remove_at_pre_s_120(ArrayList sa, ArrayList sb, int i1, int
            i2)
3644    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3645                  sa..contents = sb..contents & sa..csize = sb..csize"
3646        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3647        ensures "True" */
3648    {
3649        /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
                sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
                sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :
                sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
                sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
                sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
                <= i1 + 1 & i1 + 1 < sa..csize)" */
3650        /*: assume "0 <= i1 & i1 < sa..csize" */
3651        Object r1a = sa.remove_at(i1);
3652        /*: assume "0 <= i2 & i2 < sa..csize" */
3653        Object r2a = sa.remove_at(i2);
3654
3655        Object r2b = sb.remove_at(i2);
3656        Object r1b = sb.remove_at(i1);
3657
3658        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
3659    }
3660
3661    static void remove_at_remove_at_pre_c_120(ArrayList sa, ArrayList sb, int i1, int
            i2)
3662    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3663                  sa..contents = sb..contents & sa..csize = sb..csize"
3664        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3665        ensures "True" */
3666    {
3667        /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
                sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
                sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :
                sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
                sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
                sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
                <= i1 + 1 & i1 + 1 < sa..csize))" */
3668        /*: assume "0 <= i1 & i1 < sa..csize" */
3669        Object r1a = sa.remove_at(i1);
3670        /*: assume "0 <= i2 & i2 < sa..csize" */
3671        Object r2a = sa.remove_at(i2);
3672
3673        /*: assume "0 <= i2 & i2 < sb..csize" */
3674        Object r2b = sb.remove_at(i2);
3675        /*: assume "0 <= i1 & i1 < sb..csize" */
3676        Object r1b = sb.remove_at(i1);
3677
3678        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
3679    }
3680
3681    static void remove_at_remove_at_between_s_121(ArrayList sa, ArrayList sb, int i1,
            int i2)
3682    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
```

```
3683                 sa..contents = sb..contents & sa..csize = sb..csize"
3684         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3685         ensures "True" */
3686     {
3687         /*: assume "0 <= i1 & i1 < sa..csize" */
3688         Object r1a = sa.remove_at(i1);
3689         /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
                 sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
                 sa..csize) | (i1 = i2 & (i2, r1a) : sa..contents & 0 <= i2 & i2 < sa..csize)
                 | (sa..csize > i1 & i1 > i2 & (i1, r1a) : sa..contents & 0 <= i1 & i1 <
                 sa..csize)" */
3690         /*: assume "0 <= i2 & i2 < sa..csize" */
3691         Object r2a = sa.remove_at(i2);
3692
3693         Object r2b = sb.remove_at(i2);
3694         Object r1b = sb.remove_at(i1);
3695
3696         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                 sb..csize" */
3697     }
3698
3699     static void remove_at_remove_at_between_c_121(ArrayList sa, ArrayList sb, int i1,
            int i2)
3700     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3701                 sa..contents = sb..contents & sa..csize = sb..csize"
3702         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3703         ensures "True" */
3704     {
3705         /*: assume "0 <= i1 & i1 < sa..csize" */
3706         Object r1a = sa.remove_at(i1);
3707         /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
                 sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
                 sa..csize) | (i1 = i2 & (i2, r1a) : sa..contents & 0 <= i2 & i2 < sa..csize)
                 | (sa..csize > i1 & i1 > i2 & (i1, r1a) : sa..contents & 0 <= i1 & i1 <
                 sa..csize))" */
3708         /*: assume "0 <= i2 & i2 < sa..csize" */
3709         Object r2a = sa.remove_at(i2);
3710
3711         /*: assume "0 <= i2 & i2 < sb..csize" */
3712         Object r2b = sb.remove_at(i2);
3713         /*: assume "0 <= i1 & i1 < sb..csize" */
3714         Object r1b = sb.remove_at(i1);
3715
3716         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                 sb..csize)" */
3717     }
3718
3719     static void remove_at_remove_at_post_s_122(ArrayList sa, ArrayList sb, int i1, int
            i2)
3720     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3721                 sa..contents = sb..contents & sa..csize = sb..csize"
3722         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3723         ensures "True" */
3724     {
3725         /*: assume "0 <= i1 & i1 < sa..csize" */
3726         Object r1a = sa.remove_at(i1);
3727         /*: assume "0 <= i2 & i2 < sa..csize" */
3728         Object r2a = sa.remove_at(i2);
3729         /*: assume "(i1 < i2 & (i2 - 1, r2a) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
                 sa..csize) | (i1 = i2 & r1a = r2a) | (sa..csize + 1 > i1 & i1 > i2 & (i1 -
                 1, r1a) : sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize)" */
3730
3731         Object r2b = sb.remove_at(i2);
3732         Object r1b = sb.remove_at(i1);
3733
```

```
        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void remove_at_remove_at_post_c_122(ArrayList sa, ArrayList sb, int i1, int
        i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);
        /*: assume "~((i1 < i2 & (i2 - 1, r2a) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
            sa..csize) | (i1 = i2 & r1a = r2a) | (sa..csize + 1 > i1 & i1 > i2 & (i1 -
            1, r1a) : sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize))" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.remove_at(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void remove_at_remove_at_pre_s_123(ArrayList sa, ArrayList sb, int i1, int
        i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
            sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
            <= i1 + 1 & i1 + 1 < sa..csize)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        sb.remove_at(i2);
        Object r1b = sb.remove_at(i1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_remove_at_pre_c_123(ArrayList sa, ArrayList sb, int i1, int
        i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
```

```
            sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
            <= i1 + 1 & i1 + 1 < sa..csize))" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.remove_at(i1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void remove_at_remove_at_between_s_124(ArrayList sa, ArrayList sb, int i1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
            sa..csize) | (i1 = i2 & (i2, r1a) : sa..contents & 0 <= i2 & i2 < sa..csize)
            | (sa..csize > i1 & i1 > i2 & (i1, r1a) : sa..contents & 0 <= i1 & i1 <
            sa..csize)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        sb.remove_at(i2);
        Object r1b = sb.remove_at(i1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_remove_at_between_c_124(ArrayList sa, ArrayList sb, int i1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
            sa..csize) | (i1 = i2 & (i2, r1a) : sa..contents & 0 <= i2 & i2 < sa..csize)
            | (sa..csize > i1 & i1 > i2 & (i1, r1a) : sa..contents & 0 <= i1 & i1 <
            sa..csize))" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.remove_at(i1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }
```

```
3833    static void remove_at_remove_at_post_s_125(ArrayList sa, ArrayList sb, int i1, int
            i2)
3834    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3835                sa..contents = sb..contents & sa..csize = sb..csize"
3836        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3837        ensures "True" */
3838    {
3839        /*: assume "0 <= i1 & i1 < sa..csize" */
3840        Object r1a = sa.remove_at(i1);
3841        /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
3842        /*: ghost specvar sa__csize :: "int" = "sa..csize" */
3843        /*: assume "0 <= i2 & i2 < sa..csize" */
3844        sa.remove_at(i2);
3845        /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
                sa__contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
                sa__csize) | (i1 = i2 & (i2, r1a) : sa__contents & 0 <= i2 & i2 < sa__csize)
                | (sa..csize + 1 > i1 & i1 > i2 & (i1 - 1, r1a) : sa..contents & 0 <= i1 - 1
                & i1 - 1 < sa..csize)" */
3846
3847        sb.remove_at(i2);
3848        Object r1b = sb.remove_at(i1);
3849
3850        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
3851    }
3852
3853    static void remove_at_remove_at_post_c_125(ArrayList sa, ArrayList sb, int i1, int
            i2)
3854    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3855                sa..contents = sb..contents & sa..csize = sb..csize"
3856        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3857        ensures "True" */
3858    {
3859        /*: assume "0 <= i1 & i1 < sa..csize" */
3860        Object r1a = sa.remove_at(i1);
3861        /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
3862        /*: ghost specvar sa__csize :: "int" = "sa..csize" */
3863        /*: assume "0 <= i2 & i2 < sa..csize" */
3864        sa.remove_at(i2);
3865        /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
                sa__contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
                sa__csize) | (i1 = i2 & (i2, r1a) : sa__contents & 0 <= i2 & i2 < sa__csize)
                | (sa..csize + 1 > i1 & i1 > i2 & (i1 - 1, r1a) : sa..contents & 0 <= i1 - 1
                & i1 - 1 < sa..csize))" */
3866
3867        /*: assume "0 <= i2 & i2 < sb..csize" */
3868        sb.remove_at(i2);
3869        /*: assume "0 <= i1 & i1 < sb..csize" */
3870        Object r1b = sb.remove_at(i1);
3871
3872        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
3873    }
3874
3875    static void remove_at_set_pre_s_126(ArrayList sa, ArrayList sb, int i1, int i2,
            Object v2)
3876    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3877                sa..contents = sb..contents & sa..csize = sb..csize"
3878        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3879        ensures "True" */
3880    {
3881        /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
                sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) : sa..contents & 0
                <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & (ALL
                v. ((i1, v) : sa..contents) = ((i2 + 1, v) : sa..contents)) & (i1, v2) :
```

```
            sa..contents & (i2 + 1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <=
                i2 + 1 & i2 + 1 < sa..csize) | (i1 > i2)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        Object r2b = sb.set(i2, v2);
        Object r1b = sb.remove_at(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void remove_at_set_pre_c_126(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) : sa..contents & 0
            <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & (ALL
            v. ((i1, v) : sa..contents) = ((i2 + 1, v) : sa..contents)) & (i1, v2) :
            sa..contents & (i2 + 1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <=
            i2 + 1 & i2 + 1 < sa..csize) | (i1 > i2))" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.remove_at(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void remove_at_set_between_s_127(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa..contents & 0
            <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize) | (i1 = i2 & (i2,
            r1a) : sa..contents & r1a = v2 & (i2, v2) : sa..contents & 0 <= i2 & i2 <
            sa..csize) | i1 > i2" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        Object r2b = sb.set(i2, v2);
        Object r1b = sb.remove_at(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }
```

```
3931    static void remove_at_set_between_c_127(ArrayList sa, ArrayList sb, int i1, int i2,
            Object v2)
3932    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3933                 sa..contents = sb..contents & sa..csize = sb..csize"
3934        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3935        ensures "True" */
3936    {
3937        /*: assume "0 <= i1 & i1 < sa..csize" */
3938        Object r1a = sa.remove_at(i1);
3939        /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
                sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa..contents & 0
                <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize) | (i1 = i2 & (i2,
                r1a) : sa..contents & r1a = v2 & (i2, v2) : sa..contents & 0 <= i2 & i2 <
                sa..csize) | i1 > i2)" */
3940        /*: assume "0 <= i2 & i2 < sa..csize" */
3941        Object r2a = sa.set(i2, v2);
3942
3943        /*: assume "0 <= i2 & i2 < sb..csize" */
3944        Object r2b = sb.set(i2, v2);
3945        /*: assume "0 <= i1 & i1 < sb..csize" */
3946        Object r1b = sb.remove_at(i1);
3947
3948        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
3949    }
3950
3951    static void remove_at_set_post_s_128(ArrayList sa, ArrayList sb, int i1, int i2,
            Object v2)
3952    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3953                 sa..contents = sb..contents & sa..csize = sb..csize"
3954        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3955        ensures "True" */
3956    {
3957        /*: assume "0 <= i1 & i1 < sa..csize" */
3958        Object r1a = sa.remove_at(i1);
3959        /*: assume "0 <= i2 & i2 < sa..csize" */
3960        Object r2a = sa.set(i2, v2);
3961        /*: assume "(i1 < i2 & (i2 - 1, r2a) : sa..contents & (i2 - 1, v2) :
                sa..contents & r2a = v2 & 0 <= i2 - 1 & i2 - 1 < sa..csize) | (i1 = i2 & r1a
                = r2a & r1a = v2 & r2a = v2) | i1 > i2" */
3962
3963        Object r2b = sb.set(i2, v2);
3964        Object r1b = sb.remove_at(i1);
3965
3966        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
3967    }
3968
3969    static void remove_at_set_post_c_128(ArrayList sa, ArrayList sb, int i1, int i2,
            Object v2)
3970    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3971                 sa..contents = sb..contents & sa..csize = sb..csize"
3972        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3973        ensures "True" */
3974    {
3975        /*: assume "0 <= i1 & i1 < sa..csize" */
3976        Object r1a = sa.remove_at(i1);
3977        /*: assume "0 <= i2 & i2 < sa..csize" */
3978        Object r2a = sa.set(i2, v2);
3979        /*: assume "~((i1 < i2 & (i2 - 1, r2a) : sa..contents & (i2 - 1, v2) :
                sa..contents & r2a = v2 & 0 <= i2 - 1 & i2 - 1 < sa..csize) | (i1 = i2 & r1a
                = r2a & r1a = v2 & r2a = v2) | i1 > i2)" */
3980
3981        /*: assume "0 <= i2 & i2 < sb..csize" */
3982        Object r2b = sb.set(i2, v2);
```

```
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.remove_at(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void remove_at_set_pre_s_129(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) : sa..contents & 0
            <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & (ALL
            v. ((i1, v) : sa..contents) = ((i2 + 1, v) : sa..contents)) & (i1, v2) :
            sa..contents & (i2 + 1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <=
            i2 + 1 & i2 + 1 < sa..csize) | (i1 > i2)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        sb.set(i2, v2);
        Object r1b = sb.remove_at(i1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_set_between_s_130(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa..contents & 0
            <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize) | (i1 = i2 & (i2,
            r1a) : sa..contents & r1a = v2 & (i2, v2) : sa..contents & 0 <= i2 & i2 <
            sa..csize) | i1 > i2" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        sb.set(i2, v2);
        Object r1b = sb.remove_at(i1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_set_between_c_130(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa..contents & 0
```

```
            <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize) | (i1 = i2 & (i2,
            r1a) : sa..contents & r1a = v2 & (i2, v2) : sa..contents & 0 <= i2 & i2 <
            sa..csize) | i1 > i2)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.remove_at(i1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void remove_at_set_post_s_131(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
        /*: ghost specvar sa__csize :: "int" = "sa..csize" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);
        /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa__contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa__contents & 0
            <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa__csize) | (i1 = i2 & (i2,
            r1a) : sa__contents & r1a = v2 & (i2, v2) : sa__contents & 0 <= i2 & i2 <
            sa__csize) | i1 > i2" */

        sb.set(i2, v2);
        Object r1b = sb.remove_at(i1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_set_post_c_131(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
        /*: ghost specvar sa__csize :: "int" = "sa..csize" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);
        /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa__contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa__contents & 0
            <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa__csize) | (i1 = i2 & (i2,
            r1a) : sa__contents & r1a = v2 & (i2, v2) : sa__contents & 0 <= i2 & i2 <
            sa__csize) | i1 > i2)" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.remove_at(i1);
```

```
        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void remove_at_size_pre_s_132(ArrayList sa, ArrayList sb, int i1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                   sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "False" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        int r2a = sa.size();

        int r2b = sb.size();
        Object r1b = sb.remove_at(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void remove_at_size_pre_c_132(ArrayList sa, ArrayList sb, int i1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                   sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~(False)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        int r2a = sa.size();

        int r2b = sb.size();
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.remove_at(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void remove_at_size_between_s_133(ArrayList sa, ArrayList sb, int i1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                   sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "False" */
        int r2a = sa.size();

        int r2b = sb.size();
        Object r1b = sb.remove_at(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void remove_at_size_between_c_133(ArrayList sa, ArrayList sb, int i1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                   sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
```

```
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: assume "~(False)" */
        int r2a = sa.size();

        int r2b = sb.size();
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.remove_at(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void remove_at_size_post_s_134(ArrayList sa, ArrayList sb, int i1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        int r2a = sa.size();
        /*: assume "False" */

        int r2b = sb.size();
        Object r1b = sb.remove_at(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void remove_at_size_post_c_134(ArrayList sa, ArrayList sb, int i1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        int r2a = sa.size();
        /*: assume "~(False)" */

        int r2b = sb.size();
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.remove_at(i1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void remove_at_add_at_pre_s_135(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "(i1 < i2 & (i2, v2) : sa..contents & 0 <= i2 & i2 < sa..csize) |
            (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize) | (i1 > i2 &
            (ALL v. ((i1 - 1, v) : sa..contents) = ((i1, v) : sa..contents)) & 0 <= i1 -
            1 & i1 - 1 < sa..csize & 0 <= i1 & i1 < sa..csize)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
```

```
        sa.add_at(i2, v2);

        sb.add_at(i2, v2);
        sb.remove_at(i1);

        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_add_at_between_s_136(ArrayList sa, ArrayList sb, int i1, int
        i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "(i1 < i2 & (i2 - 1, v2) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
            sa..csize) | (i1 = i2 & (i1, v2) : sa..(old contents) & 0 <= i1 & i1 <
            sa..(old csize)) | (i1 > i2 & (ALL v. ((i1 - 1, v) : sa..contents) = ((i1,
            v) : sa..(old contents))) & 0 <= i1 - 1 & i1 - 1 < sa..csize & 0 <= i1 & i1
            < sa..(old csize))" */
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        sb.add_at(i2, v2);
        sb.remove_at(i1);

        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_add_at_post_s_137(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);
        /*: assume "(i1 < i2 & (i2 - 1, v2) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
            sa..csize) | (i1 = i2 & (i1, v2) : sa..(old contents) & 0 <= i1 & i1 <
            sa..(old csize)) | (i1 > i2 & (ALL v. ((i1, v) : sa..contents) = ((i1, v) :
            sa..(old contents))) & 0 <= i1 & i1 < sa..csize & 0 <= i1 & i1 < sa..(old
            csize))" */

        sb.add_at(i2, v2);
        sb.remove_at(i1);

        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_add_at_post_c_137(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
```

```
        sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);
        /*: assume "~((i1 < i2 & (i2 - 1, v2) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
            sa..csize) | (i1 = i2 & (i1, v2) : sa..(old contents) & 0 <= i1 & i1 <
            sa..(old csize)) | (i1 > i2 & (ALL v. ((i1, v) : sa..contents) = ((i1, v) :
            sa..(old contents))) & 0 <= i1 & i1 < sa..csize & 0 <= i1 & i1 < sa..(old
            csize)))" */

        /*: assume "0 <= i2 & i2 <= sb..csize" */
        sb.add_at(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.remove_at(i1);

        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
    }

    static void remove_at_get_pre_s_138(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | i1 > i2" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        Object r2b = sb.get(i2);
        sb.remove_at(i1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_get_pre_c_138(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | i1 > i2)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.get(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.remove_at(i1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void remove_at_get_between_s_139(ArrayList sa, ArrayList sb, int i1, int i2)
```

```
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
            sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..(old contents)) = ((i2, v) :
            sa..contents)) & 0 <= i1 & i1 < sa..(old csize) & 0 <= i2 & i2 < sa..csize)
            | i1 > i2" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        Object r2b = sb.get(i2);
        sb.remove_at(i1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_get_between_c_139(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
            sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..(old contents)) = ((i2, v) :
            sa..contents)) & 0 <= i1 & i1 < sa..(old csize) & 0 <= i2 & i2 < sa..csize)
            | i1 > i2)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.get(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.remove_at(i1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void remove_at_get_post_s_140(ArrayList sa, ArrayList sb, int i1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.get(i2);
        /*: assume "(i1 < i2 & (i2 - 1, r2a) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
            sa..csize) | (i1 = i2 & (i1, r2a) : sa..(old contents) & 0 <= i1 & i1 <
            sa..(old csize)) | i1 > i2" */

        Object r2b = sb.get(i2);
        sb.remove_at(i1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }
```

```
4359
4360    static void remove_at_get_post_c_140(ArrayList sa, ArrayList sb, int i1, int i2)
4361    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4362                sa..contents = sb..contents & sa..csize = sb..csize"
4363        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4364        ensures "True" */
4365    {
4366        /*: assume "0 <= i1 & i1 < sa..csize" */
4367        sa.remove_at(i1);
4368        /*: assume "0 <= i2 & i2 < sa..csize" */
4369        Object r2a = sa.get(i2);
4370        /*: assume "~((i1 < i2 & (i2 - 1, r2a) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
                sa..csize) | (i1 = i2 & (i1, r2a) : sa..(old contents) & 0 <= i1 & i1 <
                sa..(old csize)) | i1 > i2)" */
4371
4372        /*: assume "0 <= i2 & i2 < sb..csize" */
4373        Object r2b = sb.get(i2);
4374        /*: assume "0 <= i1 & i1 < sb..csize" */
4375        sb.remove_at(i1);
4376
4377        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
4378    }
4379
4380    static void remove_at_lastIndexOf_pre_s_144(ArrayList sa, ArrayList sb, int i1,
            Object v2)
4381    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4382                sa..contents = sb..contents & sa..csize = sb..csize"
4383        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4384        ensures "True" */
4385    {
4386        /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | ((EX i.
                (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) : sa..contents &
                i1 <= i & i < sa..csize))" */
4387        /*: assume "0 <= i1 & i1 < sa..csize" */
4388        sa.remove_at(i1);
4389        int r2a = sa.lastIndexOf(v2);
4390
4391        int r2b = sb.lastIndexOf(v2);
4392        sb.remove_at(i1);
4393
4394        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
4395    }
4396
4397    static void remove_at_lastIndexOf_pre_c_144(ArrayList sa, ArrayList sb, int i1,
            Object v2)
4398    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4399                sa..contents = sb..contents & sa..csize = sb..csize"
4400        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4401        ensures "True" */
4402    {
4403        /*: assume "~(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | ((EX
                i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
                sa..contents & i1 <= i & i < sa..csize)))" */
4404        /*: assume "0 <= i1 & i1 < sa..csize" */
4405        sa.remove_at(i1);
4406        int r2a = sa.lastIndexOf(v2);
4407
4408        int r2b = sb.lastIndexOf(v2);
4409        /*: assume "0 <= i1 & i1 < sb..csize" */
4410        sb.remove_at(i1);
4411
4412        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
4413    }
```

```
4414
4415    static void remove_at_lastIndexOf_between_c_145(ArrayList sa, ArrayList sb, int i1,
            Object v2)
4416    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4417                sa..contents = sb..contents & sa..csize = sb..csize"
4418        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4419        ensures "True" */
4420    {
4421        /*: assume "0 <= i1 & i1 < sa..csize" */
4422        sa.remove_at(i1);
4423        /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & (i1,
                v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | ((EX i. (i,
                v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) : sa..contents & i1
                <= i & i < sa..csize) & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 <
                sa..(old csize)))" */
4424        int r2a = sa.lastIndexOf(v2);
4425
4426        int r2b = sb.lastIndexOf(v2);
4427        /*: assume "0 <= i1 & i1 < sb..csize" */
4428        sb.remove_at(i1);
4429
4430        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
4431    }
4432
4433    static void remove_at_lastIndexOf_post_c_146(ArrayList sa, ArrayList sb, int i1,
            Object v2)
4434    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4435                sa..contents = sb..contents & sa..csize = sb..csize"
4436        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4437        ensures "True" */
4438    {
4439        /*: assume "0 <= i1 & i1 < sa..csize" */
4440        sa.remove_at(i1);
4441        int r2a = sa.lastIndexOf(v2);
4442        /*: assume "~((r2a < 0 & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 <
                sa..(old csize)) | (0 <= r2a & r2a < i1 & (i1, v2) ~: sa..(old contents) & 0
                <= i1 & i1 < sa..(old csize)))" */
4443
4444        int r2b = sb.lastIndexOf(v2);
4445        /*: assume "0 <= i1 & i1 < sb..csize" */
4446        sb.remove_at(i1);
4447
4448        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
4449    }
4450
4451    static void remove_at_remove_at_pre_s_147(ArrayList sa, ArrayList sb, int i1, int
            i2)
4452    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4453                sa..contents = sb..contents & sa..csize = sb..csize"
4454        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4455        ensures "True" */
4456    {
4457        /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
                sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
                sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :
                sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
                sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
                sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
                <= i1 + 1 & i1 + 1 < sa..csize)" */
4458        /*: assume "0 <= i1 & i1 < sa..csize" */
4459        sa.remove_at(i1);
4460        /*: assume "0 <= i2 & i2 < sa..csize" */
4461        Object r2a = sa.remove_at(i2);
```

```
4462
4463            Object r2b = sb.remove_at(i2);
4464            sb.remove_at(i1);
4465
4466            /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
4467        }
4468
4469        static void remove_at_remove_at_between_s_148(ArrayList sa, ArrayList sb, int i1,
                int i2)
4470        /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4471                      sa..contents = sb..contents & sa..csize = sb..csize"
4472            modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4473            ensures "True" */
4474        {
4475            /*: assume "0 <= i1 & i1 < sa..csize" */
4476            sa.remove_at(i1);
4477            /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
                    sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
                    sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..(old contents)) = ((i2, v) :
                    sa..contents)) & 0 <= i1 & i1 < sa..(old csize) & 0 <= i2 & i2 < sa..csize)
                    | (sa..csize > i1 & i1 > i2 & (ALL v. ((i1, v) : sa..(old contents)) = ((i1,
                    v) : sa..contents)) & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 & i1 <
                    sa..csize)" */
4478            /*: assume "0 <= i2 & i2 < sa..csize" */
4479            Object r2a = sa.remove_at(i2);
4480
4481            Object r2b = sb.remove_at(i2);
4482            sb.remove_at(i1);
4483
4484            /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
4485        }
4486
4487        static void remove_at_remove_at_post_s_149(ArrayList sa, ArrayList sb, int i1, int
                i2)
4488        /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4489                      sa..contents = sb..contents & sa..csize = sb..csize"
4490            modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4491            ensures "True" */
4492        {
4493            /*: assume "0 <= i1 & i1 < sa..csize" */
4494            sa.remove_at(i1);
4495            /*: assume "0 <= i2 & i2 < sa..csize" */
4496            Object r2a = sa.remove_at(i2);
4497            /*: assume "(i1 < i2 & (i2 - 1, r2a) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
                    sa..csize) | (i1 = i2 & (i1, r2a) : sa..(old contents) & 0 <= i1 & i1 <
                    sa..(old csize)) | (sa..csize + 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
                    sa..(old contents)) = ((i1 - 1, v) : sa..contents)) & 0 <= i1 & i1 <
                    sa..(old csize) & 0 <= i1 - 1 & i1 - 1 < sa..csize)" */
4498
4499            Object r2b = sb.remove_at(i2);
4500            sb.remove_at(i1);
4501
4502            /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
4503        }
4504
4505        static void remove_at_remove_at_post_c_149(ArrayList sa, ArrayList sb, int i1, int
                i2)
4506        /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4507                      sa..contents = sb..contents & sa..csize = sb..csize"
4508            modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4509            ensures "True" */
4510        {
4511            /*: assume "0 <= i1 & i1 < sa..csize" */
4512            sa.remove_at(i1);
4513            /*: assume "0 <= i2 & i2 < sa..csize" */
```

```
        Object r2a = sa.remove_at(i2);
        /*: assume "~((i1 < i2 & (i2 - 1, r2a) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
            sa..csize) | (i1 = i2 & (i1, r2a) : sa..(old contents) & 0 <= i1 & i1 <
            sa..(old csize)) | (sa..csize + 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
            sa..(old contents)) = ((i1 - 1, v) : sa..contents)) & 0 <= i1 & i1 <
            sa..(old csize) & 0 <= i1 - 1 & i1 - 1 < sa..csize))" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.remove_at(i1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void remove_at_remove_at_pre_s_150(ArrayList sa, ArrayList sb, int i1, int
        i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
            sa..csize) | i1 = i2 | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
            sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
            <= i1 + 1 & i1 + 1 < sa..csize)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        sb.remove_at(i2);
        sb.remove_at(i1);

        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_remove_at_between_s_151(ArrayList sa, ArrayList sb, int i1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
            sa..csize) | i1 = i2 | (sa..csize > i1 & i1 > i2 & (ALL v. ((i1, v) :
            sa..(old contents)) = ((i1, v) : sa..contents)) & 0 <= i1 & i1 < sa..(old
            csize) & 0 <= i1 & i1 < sa..csize)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        sb.remove_at(i2);
        sb.remove_at(i1);

        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_remove_at_post_s_152(ArrayList sa, ArrayList sb, int i1, int
        i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
```

```
4563                 sa..contents = sb..contents & sa..csize = sb..csize"
4564         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4565         ensures "True" */
4566     {
4567         /*: assume "0 <= i1 & i1 < sa..csize" */
4568         sa.remove_at(i1);
4569         /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
4570         /*: ghost specvar sa__csize :: "int" = "sa..csize" */
4571         /*: assume "0 <= i2 & i2 < sa..csize" */
4572         sa.remove_at(i2);
4573         /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
                sa__contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
                sa__csize) | i1 = i2 | (sa..csize + 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
                sa..(old contents)) = ((i1 - 1, v) : sa..contents)) & 0 <= i1 & i1 <
                sa..(old csize) & 0 <= i1 - 1 & i1 - 1 < sa..csize)" */
4574
4575         sb.remove_at(i2);
4576         sb.remove_at(i1);
4577
4578         /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
4579     }
4580
4581     static void remove_at_remove_at_post_c_152(ArrayList sa, ArrayList sb, int i1, int
            i2)
4582     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4583                 sa..contents = sb..contents & sa..csize = sb..csize"
4584         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4585         ensures "True" */
4586     {
4587         /*: assume "0 <= i1 & i1 < sa..csize" */
4588         sa.remove_at(i1);
4589         /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
4590         /*: ghost specvar sa__csize :: "int" = "sa..csize" */
4591         /*: assume "0 <= i2 & i2 < sa..csize" */
4592         sa.remove_at(i2);
4593         /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
                sa__contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
                sa__csize) | i1 = i2 | (sa..csize + 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
                sa..(old contents)) = ((i1 - 1, v) : sa..contents)) & 0 <= i1 & i1 <
                sa..(old csize) & 0 <= i1 - 1 & i1 - 1 < sa..csize))" */
4594
4595         /*: assume "0 <= i2 & i2 < sb..csize" */
4596         sb.remove_at(i2);
4597         /*: assume "0 <= i1 & i1 < sb..csize" */
4598         sb.remove_at(i1);
4599
4600         /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
4601     }
4602
4603     static void remove_at_set_pre_s_153(ArrayList sa, ArrayList sb, int i1, int i2,
            Object v2)
4604     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4605                 sa..contents = sb..contents & sa..csize = sb..csize"
4606         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4607         ensures "True" */
4608     {
4609         /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
                sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) : sa..contents & 0
                <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & (ALL
                v. ((i1, v) : sa..contents) = ((i2 + 1, v) : sa..contents)) & (i1, v2) :
                sa..contents & (i2 + 1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <=
                i2 + 1 & i2 + 1 < sa..csize) | (i1 > i2)" */
4610         /*: assume "0 <= i1 & i1 < sa..csize" */
4611         sa.remove_at(i1);
4612         /*: assume "0 <= i2 & i2 < sa..csize" */
```

```
4613        Object r2a = sa.set(i2, v2);
4614
4615        Object r2b = sb.set(i2, v2);
4616        sb.remove_at(i1);
4617
4618        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
4619    }
4620
4621    static void remove_at_set_pre_c_153(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
4622    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4623                sa..contents = sb..contents & sa..csize = sb..csize"
4624        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4625        ensures "True" */
4626    {
4627        /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) : sa..contents & 0
            <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & (ALL
            v. ((i1, v) : sa..contents) = ((i2 + 1, v) : sa..contents)) & (i1, v2) :
            sa..contents & (i2 + 1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <=
            i2 + 1 & i2 + 1 < sa..csize) | (i1 > i2))" */
4628        /*: assume "0 <= i1 & i1 < sa..csize" */
4629        sa.remove_at(i1);
4630        /*: assume "0 <= i2 & i2 < sa..csize" */
4631        Object r2a = sa.set(i2, v2);
4632
4633        /*: assume "0 <= i2 & i2 < sb..csize" */
4634        Object r2b = sb.set(i2, v2);
4635        /*: assume "0 <= i1 & i1 < sb..csize" */
4636        sb.remove_at(i1);
4637
4638        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
4639    }
4640
4641    static void remove_at_set_between_s_154(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
4642    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4643                sa..contents = sb..contents & sa..csize = sb..csize"
4644        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4645        ensures "True" */
4646    {
4647        /*: assume "0 <= i1 & i1 < sa..csize" */
4648        sa.remove_at(i1);
4649        /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa..contents & 0
            <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize) | (i1 = i2 & (ALL
            v. ((i1, v) : sa..(old contents)) = ((i2, v) : sa..contents)) & (i1, v2) :
            sa..(old contents) & (i2, v2) : sa..contents & 0 <= i1 & i1 < sa..(old
            csize) & 0 <= i2 & i2 < sa..csize) | i1 > i2" */
4650        /*: assume "0 <= i2 & i2 < sa..csize" */
4651        Object r2a = sa.set(i2, v2);
4652
4653        Object r2b = sb.set(i2, v2);
4654        sb.remove_at(i1);
4655
4656        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
4657    }
4658
4659    static void remove_at_set_between_c_154(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
4660    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4661                sa..contents = sb..contents & sa..csize = sb..csize"
4662        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4663        ensures "True" */
```

```
4664    {
4665        /*: assume "0 <= i1 & i1 < sa..csize" */
4666        sa.remove_at(i1);
4667        /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
                sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa..contents & 0
                <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize) | (i1 = i2 & (ALL
                v. ((i1, v) : sa..(old contents)) = ((i2, v) : sa..contents)) & (i1, v2) :
                sa..(old contents) & (i2, v2) : sa..contents & 0 <= i1 & i1 < sa..(old
                csize) & 0 <= i2 & i2 < sa..csize) | i1 > i2)" */
4668        /*: assume "0 <= i2 & i2 < sa..csize" */
4669        Object r2a = sa.set(i2, v2);
4670
4671        /*: assume "0 <= i2 & i2 < sb..csize" */
4672        Object r2b = sb.set(i2, v2);
4673        /*: assume "0 <= i1 & i1 < sb..csize" */
4674        sb.remove_at(i1);
4675
4676        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
4677    }
4678
4679    static void remove_at_set_post_s_155(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
4680    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4681                  sa..contents = sb..contents & sa..csize = sb..csize"
4682        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4683        ensures "True" */
4684    {
4685        /*: assume "0 <= i1 & i1 < sa..csize" */
4686        sa.remove_at(i1);
4687        /*: assume "0 <= i2 & i2 < sa..csize" */
4688        Object r2a = sa.set(i2, v2);
4689        /*: assume "(i1 < i2 & (i2 - 1, r2a) : sa..contents & (i2 - 1, v2) :
                sa..contents & r2a = v2 & 0 <= i2 - 1 & i2 - 1 < sa..csize) | (i1 = i2 &
                (i1, r2a) : sa..(old contents) & (i1, v2) : sa..(old contents) & r2a = v2 &
                0 <= i1 & i1 < sa..(old csize)) | i1 > i2" */
4690
4691        Object r2b = sb.set(i2, v2);
4692        sb.remove_at(i1);
4693
4694        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
4695    }
4696
4697    static void remove_at_set_post_c_155(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
4698    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4699                  sa..contents = sb..contents & sa..csize = sb..csize"
4700        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4701        ensures "True" */
4702    {
4703        /*: assume "0 <= i1 & i1 < sa..csize" */
4704        sa.remove_at(i1);
4705        /*: assume "0 <= i2 & i2 < sa..csize" */
4706        Object r2a = sa.set(i2, v2);
4707        /*: assume "~((i1 < i2 & (i2 - 1, r2a) : sa..contents & (i2 - 1, v2) :
                sa..contents & r2a = v2 & 0 <= i2 - 1 & i2 - 1 < sa..csize) | (i1 = i2 &
                (i1, r2a) : sa..(old contents) & (i1, v2) : sa..(old contents) & r2a = v2 &
                0 <= i1 & i1 < sa..(old csize)) | i1 > i2)" */
4708
4709        /*: assume "0 <= i2 & i2 < sb..csize" */
4710        Object r2b = sb.set(i2, v2);
4711        /*: assume "0 <= i1 & i1 < sb..csize" */
4712        sb.remove_at(i1);
4713
```

```
        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void remove_at_set_pre_s_156(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
            sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) : sa..contents & 0
            <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & (i2
            + 1, v2) : sa..contents & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 > i2)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        sb.set(i2, v2);
        sb.remove_at(i1);

        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_set_between_s_157(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa..contents & 0
            <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize) | (i1 = i2 & (i2,
            v2) : sa..contents & 0 <= i2 & i2 < sa..csize) | i1 > i2" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        sb.set(i2, v2);
        sb.remove_at(i1);

        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_set_between_c_157(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa..contents & 0
            <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize) | (i1 = i2 & (i2,
            v2) : sa..contents & 0 <= i2 & i2 < sa..csize) | i1 > i2)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
```

```
        sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.remove_at(i1);

        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
    }

    static void remove_at_set_post_s_158(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
        /*: ghost specvar sa__csize :: "int" = "sa..csize" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);
        /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa__contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa__contents & 0
            <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa__csize) | (i1 = i2 & (i2,
            v2) : sa__contents & 0 <= i2 & i2 < sa__csize) | i1 > i2" */

        sb.set(i2, v2);
        sb.remove_at(i1);

        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_set_post_c_158(ArrayList sa, ArrayList sb, int i1, int i2,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
        /*: ghost specvar sa__csize :: "int" = "sa..csize" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);
        /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa__contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa__contents & 0
            <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa__csize) | (i1 = i2 & (i2,
            v2) : sa__contents & 0 <= i2 & i2 < sa__csize) | i1 > i2)" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.remove_at(i1);

        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
    }

    static void remove_at_size_pre_s_159(ArrayList sa, ArrayList sb, int i1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "False" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
```

```
        sa.remove_at(i1);
        int r2a = sa.size();

        int r2b = sb.size();
        sb.remove_at(i1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_size_pre_c_159(ArrayList sa, ArrayList sb, int i1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~(False)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        int r2a = sa.size();

        int r2b = sb.size();
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.remove_at(i1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void remove_at_size_between_s_160(ArrayList sa, ArrayList sb, int i1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "False" */
        int r2a = sa.size();

        int r2b = sb.size();
        sb.remove_at(i1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void remove_at_size_between_c_160(ArrayList sa, ArrayList sb, int i1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "~(False)" */
        int r2a = sa.size();

        int r2b = sb.size();
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.remove_at(i1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void remove_at_size_post_s_161(ArrayList sa, ArrayList sb, int i1)
```

```
4886    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4887                sa..contents = sb..contents & sa..csize = sb..csize"
4888        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4889        ensures "True" */
4890    {
4891        /*: assume "0 <= i1 & i1 < sa..csize" */
4892        sa.remove_at(i1);
4893        int r2a = sa.size();
4894        /*: assume "False" */
4895
4896        int r2b = sb.size();
4897        sb.remove_at(i1);
4898
4899        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
4900    }
4901
4902    static void remove_at_size_post_c_161(ArrayList sa, ArrayList sb, int i1)
4903    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4904                sa..contents = sb..contents & sa..csize = sb..csize"
4905        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4906        ensures "True" */
4907    {
4908        /*: assume "0 <= i1 & i1 < sa..csize" */
4909        sa.remove_at(i1);
4910        int r2a = sa.size();
4911        /*: assume "~(False)" */
4912
4913        int r2b = sb.size();
4914        /*: assume "0 <= i1 & i1 < sb..csize" */
4915        sb.remove_at(i1);
4916
4917        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
4918    }
4919
4920    static void set_add_at_pre_s_162(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2, Object v2)
4921    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4922                sa..contents = sb..contents & sa..csize = sb..csize"
4923        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
4924        ensures "True" */
4925    {
4926        /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
                sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | (i1 > i2 & (ALL v. ((i1
                - 1, v) : sa..contents) = ((i1, v) : sa..contents)) & (i1 - 1, v1) :
                sa..contents & (i1, v1) : sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize &
                0 <= i1 & i1 < sa..csize)" */
4927        /*: assume "0 <= i1 & i1 < sa..csize" */
4928        Object r1a = sa.set(i1, v1);
4929        /*: assume "0 <= i2 & i2 <= sa..csize" */
4930        sa.add_at(i2, v2);
4931
4932        sb.add_at(i2, v2);
4933        Object r1b = sb.set(i1, v1);
4934
4935        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
4936    }
4937
4938    static void set_add_at_pre_c_162(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2, Object v2)
4939    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4940                sa..contents = sb..contents & sa..csize = sb..csize"
4941        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
```

```
        ensures "True" */
    {
        /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
            sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | (i1 > i2 & (ALL v. ((i1
            - 1, v) : sa..contents) = ((i1, v) : sa..contents)) & (i1 - 1, v1) :
            sa..contents & (i1, v1) : sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize &
            0 <= i1 & i1 < sa..csize))" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        /*: assume "0 <= i2 & i2 <= sb..csize" */
        sb.add_at(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.set(i1, v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void set_add_at_between_s_163(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | (i1 > i2 &
            (i1 - 1, r1a) : sa..contents & (i1 - 1, v1) : sa..contents & r1a = v1 & 0 <=
            i1 - 1 & i1 - 1 < sa..csize)" */
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        sb.add_at(i2, v2);
        Object r1b = sb.set(i1, v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void set_add_at_between_c_163(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | (i1 > i2 &
            (i1 - 1, r1a) : sa..contents & (i1 - 1, v1) : sa..contents & r1a = v1 & 0 <=
            i1 - 1 & i1 - 1 < sa..csize))" */
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        /*: assume "0 <= i2 & i2 <= sb..csize" */
        sb.add_at(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.set(i1, v1);
```

```
4993        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
4994    }
4995
4996    static void set_add_at_post_s_164(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2, Object v2)
4997    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4998                    sa..contents = sb..contents & sa..csize = sb..csize"
4999        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
5000        ensures "True" */
5001    {
5002        /*: assume "0 <= i1 & i1 < sa..csize" */
5003        Object r1a = sa.set(i1, v1);
5004        /*: assume "0 <= i2 & i2 <= sa..csize" */
5005        sa.add_at(i2, v2);
5006        /*: assume "i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | (i1 > i2 &
                (i1, r1a) : sa..contents & (i1, v1) : sa..contents & r1a = v1 & 0 <= i1 & i1
                < sa..csize)" */
5007
5008        sb.add_at(i2, v2);
5009        Object r1b = sb.set(i1, v1);
5010
5011        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
5012    }
5013
5014    static void set_add_at_post_c_164(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2, Object v2)
5015    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5016                    sa..contents = sb..contents & sa..csize = sb..csize"
5017        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
5018        ensures "True" */
5019    {
5020        /*: assume "0 <= i1 & i1 < sa..csize" */
5021        Object r1a = sa.set(i1, v1);
5022        /*: assume "0 <= i2 & i2 <= sa..csize" */
5023        sa.add_at(i2, v2);
5024        /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | (i1 > i2 &
                (i1, r1a) : sa..contents & (i1, v1) : sa..contents & r1a = v1 & 0 <= i1 & i1
                < sa..csize))" */
5025
5026        /*: assume "0 <= i2 & i2 <= sb..csize" */
5027        sb.add_at(i2, v2);
5028        /*: assume "0 <= i1 & i1 < sb..csize" */
5029        Object r1b = sb.set(i1, v1);
5030
5031        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
5032    }
5033
5034    static void set_get_pre_s_165(ArrayList sa, ArrayList sb, int i1, Object v1, int i2)
5035    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5036                    sa..contents = sb..contents & sa..csize = sb..csize"
5037        modifies "sa..contents", "sb..contents"
5038        ensures "True" */
5039    {
5040        /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & 0 <= i1 & i1 <
                sa..csize) | i1 > i2" */
5041        /*: assume "0 <= i1 & i1 < sa..csize" */
5042        Object r1a = sa.set(i1, v1);
5043        /*: assume "0 <= i2 & i2 < sa..csize" */
5044        Object r2a = sa.get(i2);
5045
5046        Object r2b = sb.get(i2);
```

```
5047        Object r1b = sb.set(i1, v1);
5048
5049        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
5050    }
5051
5052    static void set_get_pre_c_165(ArrayList sa, ArrayList sb, int i1, Object v1, int i2)
5053    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5054                  sa..contents = sb..contents & sa..csize = sb..csize"
5055        modifies "sa..contents", "sb..contents"
5056        ensures "True" */
5057    {
5058        /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & 0 <= i1 & i1 <
                sa..csize) | i1 > i2)" */
5059        /*: assume "0 <= i1 & i1 < sa..csize" */
5060        Object r1a = sa.set(i1, v1);
5061        /*: assume "0 <= i2 & i2 < sa..csize" */
5062        Object r2a = sa.get(i2);
5063
5064        /*: assume "0 <= i2 & i2 < sb..csize" */
5065        Object r2b = sb.get(i2);
5066        /*: assume "0 <= i1 & i1 < sb..csize" */
5067        Object r1b = sb.set(i1, v1);
5068
5069        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
5070    }
5071
5072    static void set_get_between_s_166(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2)
5073    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5074                  sa..contents = sb..contents & sa..csize = sb..csize"
5075        modifies "sa..contents", "sb..contents"
5076        ensures "True" */
5077    {
5078        /*: assume "0 <= i1 & i1 < sa..csize" */
5079        Object r1a = sa.set(i1, v1);
5080        /*: assume "i1 < i2 | (i1 = i2 & r1a = v1) | i1 > i2" */
5081        /*: assume "0 <= i2 & i2 < sa..csize" */
5082        Object r2a = sa.get(i2);
5083
5084        Object r2b = sb.get(i2);
5085        Object r1b = sb.set(i1, v1);
5086
5087        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
5088    }
5089
5090    static void set_get_between_c_166(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2)
5091    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5092                  sa..contents = sb..contents & sa..csize = sb..csize"
5093        modifies "sa..contents", "sb..contents"
5094        ensures "True" */
5095    {
5096        /*: assume "0 <= i1 & i1 < sa..csize" */
5097        Object r1a = sa.set(i1, v1);
5098        /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v1) | i1 > i2)" */
5099        /*: assume "0 <= i2 & i2 < sa..csize" */
5100        Object r2a = sa.get(i2);
5101
5102        /*: assume "0 <= i2 & i2 < sb..csize" */
5103        Object r2b = sb.get(i2);
5104        /*: assume "0 <= i1 & i1 < sb..csize" */
5105        Object r1b = sb.set(i1, v1);
```

```
5106
5107        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
5108    }
5109
5110    static void set_get_post_s_167(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2)
5111    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5112                  sa..contents = sb..contents & sa..csize = sb..csize"
5113        modifies "sa..contents", "sb..contents"
5114        ensures "True" */
5115    {
5116        /*: assume "0 <= i1 & i1 < sa..csize" */
5117        Object r1a = sa.set(i1, v1);
5118        /*: assume "0 <= i2 & i2 < sa..csize" */
5119        Object r2a = sa.get(i2);
5120        /*: assume "i1 < i2 | (i1 = i2 & r1a = v1) | i1 > i2" */
5121
5122        Object r2b = sb.get(i2);
5123        Object r1b = sb.set(i1, v1);
5124
5125        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
5126    }
5127
5128    static void set_get_post_c_167(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2)
5129    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5130                  sa..contents = sb..contents & sa..csize = sb..csize"
5131        modifies "sa..contents", "sb..contents"
5132        ensures "True" */
5133    {
5134        /*: assume "0 <= i1 & i1 < sa..csize" */
5135        Object r1a = sa.set(i1, v1);
5136        /*: assume "0 <= i2 & i2 < sa..csize" */
5137        Object r2a = sa.get(i2);
5138        /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v1) | i1 > i2)" */
5139
5140        /*: assume "0 <= i2 & i2 < sb..csize" */
5141        Object r2b = sb.get(i2);
5142        /*: assume "0 <= i1 & i1 < sb..csize" */
5143        Object r1b = sb.set(i1, v1);
5144
5145        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
5146    }
5147
5148    static void set_indexOf_pre_s_168(ArrayList sa, ArrayList sb, int i1, Object v1,
            Object v2)
5149    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5150                  sa..contents = sb..contents & sa..csize = sb..csize"
5151        modifies "sa..contents", "sb..contents"
5152        ensures "True" */
5153    {
5154        /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
                v2) | (EX i. (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) :
                sa..contents & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 <
                sa..csize & v1 = v2) | (~(EX i. (i, v2) : sa..contents & 0 <= i & i <= i1) &
                (EX i. (i, v2) : sa..contents & i1 < i & i < sa..csize) & v1 ~= v2)" */
5155        /*: assume "0 <= i1 & i1 < sa..csize" */
5156        Object r1a = sa.set(i1, v1);
5157        int r2a = sa.indexOf(v2);
5158
5159        int r2b = sb.indexOf(v2);
5160        Object r1b = sb.set(i1, v1);
```

```
5161
5162        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
5163    }
5164
5165    static void set_indexOf_pre_c_168(ArrayList sa, ArrayList sb, int i1, Object v1,
            Object v2)
5166    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5167                  sa..contents = sb..contents & sa..csize = sb..csize"
5168        modifies "sa..contents", "sb..contents"
5169        ensures "True" */
5170    {
5171        /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
                v2) | (EX i. (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) :
                sa..contents & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 <
                sa..csize & v1 = v2) | (~(EX i. (i, v2) : sa..contents & 0 <= i & i <= i1) &
                (EX i. (i, v2) : sa..contents & i1 < i & i < sa..csize) & v1 ~= v2))" */
5172        /*: assume "0 <= i1 & i1 < sa..csize" */
5173        Object r1a = sa.set(i1, v1);
5174        int r2a = sa.indexOf(v2);
5175
5176        int r2b = sb.indexOf(v2);
5177        /*: assume "0 <= i1 & i1 < sb..csize" */
5178        Object r1b = sb.set(i1, v1);
5179
5180        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
5181    }
5182
5183    static void set_indexOf_between_s_169(ArrayList sa, ArrayList sb, int i1, Object
            v1, Object v2)
5184    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5185                  sa..contents = sb..contents & sa..csize = sb..csize"
5186        modifies "sa..contents", "sb..contents"
5187        ensures "True" */
5188    {
5189        /*: assume "0 <= i1 & i1 < sa..csize" */
5190        Object r1a = sa.set(i1, v1);
5191        /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & r1a ~=
                v2) | (EX i. (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) :
                sa..contents & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 <
                sa..csize & r1a = v2) | (~(EX i. (i, v2) : sa..contents & 0 <= i & i <= i1)
                & (EX i. (i, v2) : sa..contents & i1 < i & i < sa..csize) & r1a ~= v2)" */
5192        int r2a = sa.indexOf(v2);
5193
5194        int r2b = sb.indexOf(v2);
5195        Object r1b = sb.set(i1, v1);
5196
5197        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
5198    }
5199
5200    static void set_indexOf_between_c_169(ArrayList sa, ArrayList sb, int i1, Object
            v1, Object v2)
5201    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5202                  sa..contents = sb..contents & sa..csize = sb..csize"
5203        modifies "sa..contents", "sb..contents"
5204        ensures "True" */
5205    {
5206        /*: assume "0 <= i1 & i1 < sa..csize" */
5207        Object r1a = sa.set(i1, v1);
5208        /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & r1a
                ~= v2) | (EX i. (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2)
                : sa..contents & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 <
```

```
            sa..csize & r1a = v2) | (~(EX i. (i, v2) : sa..contents & 0 <= i & i <= i1)
            & (EX i. (i, v2) : sa..contents & i1 < i & i < sa..csize) & r1a ~= v2))" */
5209        int r2a = sa.indexOf(v2);
5210
5211        int r2b = sb.indexOf(v2);
5212        /*: assume "0 <= i1 & i1 < sb..csize" */
5213        Object r1b = sb.set(i1, v1);
5214
5215        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
5216    }
5217
5218    static void set_indexOf_post_s_170(ArrayList sa, ArrayList sb, int i1, Object v1,
        Object v2)
5219    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5220                  sa..contents = sb..contents & sa..csize = sb..csize"
5221        modifies "sa..contents", "sb..contents"
5222        ensures "True" */
5223    {
5224        /*: assume "0 <= i1 & i1 < sa..csize" */
5225        Object r1a = sa.set(i1, v1);
5226        int r2a = sa.indexOf(v2);
5227        /*: assume "(r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1) | (r2a = i1 & r1a =
                v2) | (r2a > i1 & r1a ~= v2)" */
5228
5229        int r2b = sb.indexOf(v2);
5230        Object r1b = sb.set(i1, v1);
5231
5232        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
5233    }
5234
5235    static void set_indexOf_post_c_170(ArrayList sa, ArrayList sb, int i1, Object v1,
        Object v2)
5236    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5237                  sa..contents = sb..contents & sa..csize = sb..csize"
5238        modifies "sa..contents", "sb..contents"
5239        ensures "True" */
5240    {
5241        /*: assume "0 <= i1 & i1 < sa..csize" */
5242        Object r1a = sa.set(i1, v1);
5243        int r2a = sa.indexOf(v2);
5244        /*: assume "~((r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1) | (r2a = i1 & r1a =
                v2) | (r2a > i1 & r1a ~= v2))" */
5245
5246        int r2b = sb.indexOf(v2);
5247        /*: assume "0 <= i1 & i1 < sb..csize" */
5248        Object r1b = sb.set(i1, v1);
5249
5250        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
5251    }
5252
5253    static void set_lastIndexOf_pre_s_171(ArrayList sa, ArrayList sb, int i1, Object
        v1, Object v2)
5254    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5255                  sa..contents = sb..contents & sa..csize = sb..csize"
5256        modifies "sa..contents", "sb..contents"
5257        ensures "True" */
5258    {
5259        /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
                v2) | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
                sa..contents & i1 <= i & i < sa..csize) & v1 ~= v2) | ((i1, v2) :
                sa..contents & 0 <= i1 & i1 < sa..csize & ~(EX i. (i, v2) : sa..contents &
```

```
        i1 < i & i < sa..csize) & v1 = v2) | (EX i. (i, v2) : sa..contents & i1 < i
        & i < sa..csize)" */
    /*: assume "0 <= i1 & i1 < sa..csize" */
    Object r1a = sa.set(i1, v1);
    int r2a = sa.lastIndexOf(v2);

    int r2b = sb.lastIndexOf(v2);
    Object r1b = sb.set(i1, v1);

    /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
}

static void set_lastIndexOf_pre_c_171(ArrayList sa, ArrayList sb, int i1, Object
    v1, Object v2)
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
             sa..contents = sb..contents & sa..csize = sb..csize"
    modifies "sa..contents", "sb..contents"
    ensures "True" */
{
    /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
        v2) | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
        sa..contents & i1 <= i & i < sa..csize) & v1 ~= v2) | ((i1, v2) :
        sa..contents & 0 <= i1 & i1 < sa..csize & ~(EX i. (i, v2) : sa..contents &
        i1 < i & i < sa..csize) & v1 = v2) | (EX i. (i, v2) : sa..contents & i1 < i
        & i < sa..csize))" */
    /*: assume "0 <= i1 & i1 < sa..csize" */
    Object r1a = sa.set(i1, v1);
    int r2a = sa.lastIndexOf(v2);

    int r2b = sb.lastIndexOf(v2);
    /*: assume "0 <= i1 & i1 < sb..csize" */
    Object r1b = sb.set(i1, v1);

    /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
}

static void set_lastIndexOf_between_s_172(ArrayList sa, ArrayList sb, int i1,
    Object v1, Object v2)
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
             sa..contents = sb..contents & sa..csize = sb..csize"
    modifies "sa..contents", "sb..contents"
    ensures "True" */
{
    /*: assume "0 <= i1 & i1 < sa..csize" */
    Object r1a = sa.set(i1, v1);
    /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & r1a ~=
        v2) | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
        sa..contents & i1 <= i & i < sa..csize) & r1a ~= v2) | ((i1, v2) :
        sa..contents & 0 <= i1 & i1 < sa..csize & ~(EX i. (i, v2) : sa..contents &
        i1 < i & i < sa..csize) & r1a = v2) | (EX i. (i, v2) : sa..contents & i1 < i
        & i < sa..csize)" */
    int r2a = sa.lastIndexOf(v2);

    int r2b = sb.lastIndexOf(v2);
    Object r1b = sb.set(i1, v1);

    /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
}

static void set_lastIndexOf_between_c_172(ArrayList sa, ArrayList sb, int i1,
    Object v1, Object v2)
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
```

```
5307                 sa..contents = sb..contents & sa..csize = sb..csize"
5308         modifies "sa..contents", "sb..contents"
5309         ensures "True" */
5310     {
5311         /*: assume "0 <= i1 & i1 < sa..csize" */
5312         Object r1a = sa.set(i1, v1);
5313         /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & r1a
                 ~= v2) | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2)
                 : sa..contents & i1 <= i & i < sa..csize) & r1a ~= v2) | ((i1, v2) :
                 sa..contents & 0 <= i1 & i1 < sa..csize & ~(EX i. (i, v2) : sa..contents &
                 i1 < i & i < sa..csize) & r1a = v2) | (EX i. (i, v2) : sa..contents & i1 < i
                 & i < sa..csize))" */
5314         int r2a = sa.lastIndexOf(v2);
5315 
5316         int r2b = sb.lastIndexOf(v2);
5317         /*: assume "0 <= i1 & i1 < sb..csize" */
5318         Object r1b = sb.set(i1, v1);
5319 
5320         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                 sb..csize)" */
5321     }
5322 
5323     static void set_lastIndexOf_post_s_173(ArrayList sa, ArrayList sb, int i1, Object
            v1, Object v2)
5324     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5325                 sa..contents = sb..contents & sa..csize = sb..csize"
5326         modifies "sa..contents", "sb..contents"
5327         ensures "True" */
5328     {
5329         /*: assume "0 <= i1 & i1 < sa..csize" */
5330         Object r1a = sa.set(i1, v1);
5331         int r2a = sa.lastIndexOf(v2);
5332         /*: assume "(r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1 & r1a ~= v2) | (r2a =
                 i1 & r1a = v2) | r2a > i1" */
5333 
5334         int r2b = sb.lastIndexOf(v2);
5335         Object r1b = sb.set(i1, v1);
5336 
5337         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                 sb..csize" */
5338     }
5339 
5340     static void set_lastIndexOf_post_c_173(ArrayList sa, ArrayList sb, int i1, Object
            v1, Object v2)
5341     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5342                 sa..contents = sb..contents & sa..csize = sb..csize"
5343         modifies "sa..contents", "sb..contents"
5344         ensures "True" */
5345     {
5346         /*: assume "0 <= i1 & i1 < sa..csize" */
5347         Object r1a = sa.set(i1, v1);
5348         int r2a = sa.lastIndexOf(v2);
5349         /*: assume "~((r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1 & r1a ~= v2) | (r2a
                 = i1 & r1a = v2) | r2a > i1)" */
5350 
5351         int r2b = sb.lastIndexOf(v2);
5352         /*: assume "0 <= i1 & i1 < sb..csize" */
5353         Object r1b = sb.set(i1, v1);
5354 
5355         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                 sb..csize)" */
5356     }
5357 
5358     static void set_remove_at_pre_s_174(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2)
```

```
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (ALL v. ((i1, v) :
            sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
            (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
            1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
            sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
            (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
            1 < sa..csize)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);

        Object r2b = sb.remove_at(i2);
        Object r1b = sb.set(i1, v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void set_remove_at_pre_c_174(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (ALL v. ((i1, v) :
            sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
            (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
            1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
            sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
            (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
            1 < sa..csize))" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.set(i1, v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void set_remove_at_between_s_175(ArrayList sa, ArrayList sb, int i1, Object
        v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, r1a) :
            sa..contents & r1a = v1 & (i1 + 1, v1) : sa..contents & 0 <= i1 + 1 & i1 + 1
            < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (i1 + 1, r1a) : sa..contents
```

```
            & r1a = v1 & (i1 + 1, v1) : sa..contents & 0 <= i1 + 1 & i1 + 1 <
                sa..csize)" */
5405        /*: assume "0 <= i2 & i2 < sa..csize" */
5406        Object r2a = sa.remove_at(i2);
5407
5408        Object r2b = sb.remove_at(i2);
5409        Object r1b = sb.set(i1, v1);
5410
5411        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
5412    }
5413
5414    static void set_remove_at_between_c_175(ArrayList sa, ArrayList sb, int i1, Object
           v1, int i2)
5415    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5416                  sa..contents = sb..contents & sa..csize = sb..csize"
5417        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5418        ensures "True" */
5419    {
5420        /*: assume "0 <= i1 & i1 < sa..csize" */
5421        Object r1a = sa.set(i1, v1);
5422        /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, r1a) :
                sa..contents & r1a = v1 & (i1 + 1, v1) : sa..contents & 0 <= i1 + 1 & i1 + 1
                < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (i1 + 1, r1a) : sa..contents
                & r1a = v1 & (i1 + 1, v1) : sa..contents & 0 <= i1 + 1 & i1 + 1 <
                sa..csize))" */
5423        /*: assume "0 <= i2 & i2 < sa..csize" */
5424        Object r2a = sa.remove_at(i2);
5425
5426        /*: assume "0 <= i2 & i2 < sb..csize" */
5427        Object r2b = sb.remove_at(i2);
5428        /*: assume "0 <= i1 & i1 < sb..csize" */
5429        Object r1b = sb.set(i1, v1);
5430
5431        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
5432    }
5433
5434    static void set_remove_at_post_s_176(ArrayList sa, ArrayList sb, int i1, Object v1,
           int i2)
5435    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5436                  sa..contents = sb..contents & sa..csize = sb..csize"
5437        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5438        ensures "True" */
5439    {
5440        /*: assume "0 <= i1 & i1 < sa..csize" */
5441        Object r1a = sa.set(i1, v1);
5442        /*: assume "0 <= i2 & i2 < sa..csize" */
5443        Object r2a = sa.remove_at(i2);
5444        /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize & (i1, r1a) : sa..contents &
                r1a = v1 & (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize) | (sa..csize
                > i1 & i1 > i2 & (i1, r1a) : sa..contents & r1a = v1 & (i1, v1) :
                sa..contents & 0 <= i1 & i1 < sa..csize)" */
5445
5446        Object r2b = sb.remove_at(i2);
5447        Object r1b = sb.set(i1, v1);
5448
5449        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
5450    }
5451
5452    static void set_remove_at_post_c_176(ArrayList sa, ArrayList sb, int i1, Object v1,
           int i2)
5453    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5454                  sa..contents = sb..contents & sa..csize = sb..csize"
```

```
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);
        /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize & (i1, r1a) : sa..contents &
            r1a = v1 & (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize) | (sa..csize
            > i1 & i1 > i2 & (i1, r1a) : sa..contents & r1a = v1 & (i1, v1) :
            sa..contents & 0 <= i1 & i1 < sa..csize))" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.set(i1, v1);

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void set_remove_at_pre_s_177(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (ALL v. ((i1, v) :
            sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
            (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
            1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
            sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
            (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
            1 < sa..csize)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        sb.remove_at(i2);
        Object r1b = sb.set(i1, v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void set_remove_at_pre_c_177(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (ALL v. ((i1, v) :
            sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
            (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
            1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
            sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
            (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
            1 < sa..csize))" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

```

```
        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.set(i1, v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void set_remove_at_between_s_178(ArrayList sa, ArrayList sb, int i1, Object
        v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, r1a) :
            sa..contents & r1a = v1 & (i1 + 1, v1) : sa..contents & 0 <= i1 + 1 & i1 + 1
            < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (i1 + 1, r1a) : sa..contents
            & r1a = v1 & (i1 + 1, v1) : sa..contents & 0 <= i1 + 1 & i1 + 1 <
            sa..csize)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        sb.remove_at(i2);
        Object r1b = sb.set(i1, v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void set_remove_at_between_c_178(ArrayList sa, ArrayList sb, int i1, Object
        v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, r1a) :
            sa..contents & r1a = v1 & (i1 + 1, v1) : sa..contents & 0 <= i1 + 1 & i1 + 1
            < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (i1 + 1, r1a) : sa..contents
            & r1a = v1 & (i1 + 1, v1) : sa..contents & 0 <= i1 + 1 & i1 + 1 <
            sa..csize))" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.set(i1, v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void set_remove_at_post_s_179(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
```

```
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);
        /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize & (i1, r1a) : sa..contents &
            r1a = v1 & (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize) | (sa..csize
            > i1 & i1 > i2 & (i1, r1a) : sa..contents & r1a = v1 & (i1, v1) :
            sa..contents & 0 <= i1 & i1 < sa..csize)" */

        sb.remove_at(i2);
        Object r1b = sb.set(i1, v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void set_remove_at_post_c_179(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);
        /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize & (i1, r1a) : sa..contents &
            r1a = v1 & (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize) | (sa..csize
            > i1 & i1 > i2 & (i1, r1a) : sa..contents & r1a = v1 & (i1, v1) :
            sa..contents & 0 <= i1 & i1 < sa..csize))" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.set(i1, v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void set_set_pre_s_180(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
            sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | i1 > i2" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        Object r2b = sb.set(i2, v2);
        Object r1b = sb.set(i1, v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void set_set_pre_c_180(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
```

```
5607        modifies "sa..contents", "sb..contents"
5608        ensures "True" */
5609    {
5610        /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
                sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | i1 > i2)" */
5611        /*: assume "0 <= i1 & i1 < sa..csize" */
5612        Object r1a = sa.set(i1, v1);
5613        /*: assume "0 <= i2 & i2 < sa..csize" */
5614        Object r2a = sa.set(i2, v2);
5615
5616        /*: assume "0 <= i2 & i2 < sb..csize" */
5617        Object r2b = sb.set(i2, v2);
5618        /*: assume "0 <= i1 & i1 < sb..csize" */
5619        Object r1b = sb.set(i1, v1);
5620
5621        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
5622    }
5623
5624    static void set_set_between_s_181(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2, Object v2)
5625    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5626                  sa..contents = sb..contents & sa..csize = sb..csize"
5627        modifies "sa..contents", "sb..contents"
5628        ensures "True" */
5629    {
5630        /*: assume "0 <= i1 & i1 < sa..csize" */
5631        Object r1a = sa.set(i1, v1);
5632        /*: assume "i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | i1 > i2" */
5633        /*: assume "0 <= i2 & i2 < sa..csize" */
5634        Object r2a = sa.set(i2, v2);
5635
5636        Object r2b = sb.set(i2, v2);
5637        Object r1b = sb.set(i1, v1);
5638
5639        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
5640    }
5641
5642    static void set_set_between_c_181(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2, Object v2)
5643    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5644                  sa..contents = sb..contents & sa..csize = sb..csize"
5645        modifies "sa..contents", "sb..contents"
5646        ensures "True" */
5647    {
5648        /*: assume "0 <= i1 & i1 < sa..csize" */
5649        Object r1a = sa.set(i1, v1);
5650        /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | i1 > i2)" */
5651        /*: assume "0 <= i2 & i2 < sa..csize" */
5652        Object r2a = sa.set(i2, v2);
5653
5654        /*: assume "0 <= i2 & i2 < sb..csize" */
5655        Object r2b = sb.set(i2, v2);
5656        /*: assume "0 <= i1 & i1 < sb..csize" */
5657        Object r1b = sb.set(i1, v1);
5658
5659        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
5660    }
5661
5662    static void set_set_post_s_182(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2, Object v2)
5663    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5664                  sa..contents = sb..contents & sa..csize = sb..csize"
```

```
5665        modifies "sa..contents", "sb..contents"
5666        ensures "True" */
5667    {
5668        /*: assume "0 <= i1 & i1 < sa..csize" */
5669        Object r1a = sa.set(i1, v1);
5670        /*: assume "0 <= i2 & i2 < sa..csize" */
5671        Object r2a = sa.set(i2, v2);
5672        /*: assume "i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | i1 > i2" */
5673
5674        Object r2b = sb.set(i2, v2);
5675        Object r1b = sb.set(i1, v1);
5676
5677        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
5678    }
5679
5680    static void set_set_post_c_182(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2, Object v2)
5681    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5682                  sa..contents = sb..contents & sa..csize = sb..csize"
5683        modifies "sa..contents", "sb..contents"
5684        ensures "True" */
5685    {
5686        /*: assume "0 <= i1 & i1 < sa..csize" */
5687        Object r1a = sa.set(i1, v1);
5688        /*: assume "0 <= i2 & i2 < sa..csize" */
5689        Object r2a = sa.set(i2, v2);
5690        /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | i1 > i2)" */
5691
5692        /*: assume "0 <= i2 & i2 < sb..csize" */
5693        Object r2b = sb.set(i2, v2);
5694        /*: assume "0 <= i1 & i1 < sb..csize" */
5695        Object r1b = sb.set(i1, v1);
5696
5697        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
5698    }
5699
5700    static void set_set_pre_s_183(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2, Object v2)
5701    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5702                  sa..contents = sb..contents & sa..csize = sb..csize"
5703        modifies "sa..contents", "sb..contents"
5704        ensures "True" */
5705    {
5706        /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
                sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | i1 > i2" */
5707        /*: assume "0 <= i1 & i1 < sa..csize" */
5708        Object r1a = sa.set(i1, v1);
5709        /*: assume "0 <= i2 & i2 < sa..csize" */
5710        sa.set(i2, v2);
5711
5712        sb.set(i2, v2);
5713        Object r1b = sb.set(i1, v1);
5714
5715        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
5716    }
5717
5718    static void set_set_pre_c_183(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2, Object v2)
5719    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5720                  sa..contents = sb..contents & sa..csize = sb..csize"
5721        modifies "sa..contents", "sb..contents"
5722        ensures "True" */
5723    {
```

```
        /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
            sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | i1 > i2)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.set(i1, v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void set_set_between_s_184(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | i1 > i2" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        sb.set(i2, v2);
        Object r1b = sb.set(i1, v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void set_set_between_c_184(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.set(i1, v1);
        /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | i1 > i2)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.set(i1, v1);

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void set_set_post_s_185(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
```

```
5783        Object r1a = sa.set(i1, v1);
5784        /*: assume "0 <= i2 & i2 < sa..csize" */
5785        sa.set(i2, v2);
5786        /*: assume "i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | i1 > i2" */
5787
5788        sb.set(i2, v2);
5789        Object r1b = sb.set(i1, v1);
5790
5791        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
5792    }
5793
5794    static void set_set_post_c_185(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
5795    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5796                  sa..contents = sb..contents & sa..csize = sb..csize"
5797        modifies "sa..contents", "sb..contents"
5798        ensures "True" */
5799    {
5800        /*: assume "0 <= i1 & i1 < sa..csize" */
5801        Object r1a = sa.set(i1, v1);
5802        /*: assume "0 <= i2 & i2 < sa..csize" */
5803        sa.set(i2, v2);
5804        /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | i1 > i2)" */
5805
5806        /*: assume "0 <= i2 & i2 < sb..csize" */
5807        sb.set(i2, v2);
5808        /*: assume "0 <= i1 & i1 < sb..csize" */
5809        Object r1b = sb.set(i1, v1);
5810
5811        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
5812    }
5813
5814    static void set_size_pre_s_186(ArrayList sa, ArrayList sb, int i1, Object v1)
5815    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5816                  sa..contents = sb..contents & sa..csize = sb..csize"
5817        modifies "sa..contents", "sb..contents"
5818        ensures "True" */
5819    {
5820        /*: assume "True" */
5821        /*: assume "0 <= i1 & i1 < sa..csize" */
5822        Object r1a = sa.set(i1, v1);
5823        int r2a = sa.size();
5824
5825        int r2b = sb.size();
5826        Object r1b = sb.set(i1, v1);
5827
5828        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
5829    }
5830
5831    static void set_size_pre_c_186(ArrayList sa, ArrayList sb, int i1, Object v1)
5832    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5833                  sa..contents = sb..contents & sa..csize = sb..csize"
5834        modifies "sa..contents", "sb..contents"
5835        ensures "True" */
5836    {
5837        /*: assume "~(True)" */
5838        /*: assume "0 <= i1 & i1 < sa..csize" */
5839        Object r1a = sa.set(i1, v1);
5840        int r2a = sa.size();
5841
5842        int r2b = sb.size();
5843        /*: assume "0 <= i1 & i1 < sb..csize" */
5844        Object r1b = sb.set(i1, v1);
```

```
5845
5846        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
5847    }
5848
5849    static void set_size_between_s_187(ArrayList sa, ArrayList sb, int i1, Object v1)
5850    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5851                  sa..contents = sb..contents & sa..csize = sb..csize"
5852        modifies "sa..contents", "sb..contents"
5853        ensures "True" */
5854    {
5855        /*: assume "0 <= i1 & i1 < sa..csize" */
5856        Object r1a = sa.set(i1, v1);
5857        /*: assume "True" */
5858        int r2a = sa.size();
5859
5860        int r2b = sb.size();
5861        Object r1b = sb.set(i1, v1);
5862
5863        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
5864    }
5865
5866    static void set_size_between_c_187(ArrayList sa, ArrayList sb, int i1, Object v1)
5867    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5868                  sa..contents = sb..contents & sa..csize = sb..csize"
5869        modifies "sa..contents", "sb..contents"
5870        ensures "True" */
5871    {
5872        /*: assume "0 <= i1 & i1 < sa..csize" */
5873        Object r1a = sa.set(i1, v1);
5874        /*: assume "~(True)" */
5875        int r2a = sa.size();
5876
5877        int r2b = sb.size();
5878        /*: assume "0 <= i1 & i1 < sb..csize" */
5879        Object r1b = sb.set(i1, v1);
5880
5881        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
5882    }
5883
5884    static void set_size_post_s_188(ArrayList sa, ArrayList sb, int i1, Object v1)
5885    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5886                  sa..contents = sb..contents & sa..csize = sb..csize"
5887        modifies "sa..contents", "sb..contents"
5888        ensures "True" */
5889    {
5890        /*: assume "0 <= i1 & i1 < sa..csize" */
5891        Object r1a = sa.set(i1, v1);
5892        int r2a = sa.size();
5893        /*: assume "True" */
5894
5895        int r2b = sb.size();
5896        Object r1b = sb.set(i1, v1);
5897
5898        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
5899    }
5900
5901    static void set_size_post_c_188(ArrayList sa, ArrayList sb, int i1, Object v1)
5902    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5903                  sa..contents = sb..contents & sa..csize = sb..csize"
5904        modifies "sa..contents", "sb..contents"
5905        ensures "True" */
```

```
5906    {
5907        /*: assume "0 <= i1 & i1 < sa..csize" */
5908        Object r1a = sa.set(i1, v1);
5909        int r2a = sa.size();
5910        /*: assume "~(True)" */
5911
5912        int r2b = sb.size();
5913        /*: assume "0 <= i1 & i1 < sb..csize" */
5914        Object r1b = sb.set(i1, v1);
5915
5916        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
5917    }
5918
5919    static void set_add_at_pre_s_189(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2, Object v2)
5920    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5921                  sa..contents = sb..contents & sa..csize = sb..csize"
5922        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
5923        ensures "True" */
5924    {
5925        /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
                sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | (i1 > i2 & (ALL v. ((i1
                - 1, v) : sa..contents) = ((i1, v) : sa..contents)) & (i1 - 1, v1) :
                sa..contents & (i1, v1) : sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize &
                0 <= i1 & i1 < sa..csize)" */
5926        /*: assume "0 <= i1 & i1 < sa..csize" */
5927        sa.set(i1, v1);
5928        /*: assume "0 <= i2 & i2 <= sa..csize" */
5929        sa.add_at(i2, v2);
5930
5931        sb.add_at(i2, v2);
5932        sb.set(i1, v1);
5933
5934        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
5935    }
5936
5937    static void set_add_at_between_s_190(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2, Object v2)
5938    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5939                  sa..contents = sb..contents & sa..csize = sb..csize"
5940        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
5941        ensures "True" */
5942    {
5943        /*: assume "0 <= i1 & i1 < sa..csize" */
5944        sa.set(i1, v1);
5945        /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & (i1, v2) :
                sa..(old contents) & v1 = v2 & 0 <= i1 & i1 < sa..(old csize)) | (i1 > i2 &
                (ALL v. ((i1 - 1, v) : sa..contents) = ((i1, v) : sa..(old contents))) & (i1
                - 1, v1) : sa..contents & (i1, v1) : sa..(old contents) & 0 <= i1 - 1 & i1 -
                1 < sa..csize & 0 <= i1 & i1 < sa..(old csize))" */
5946        /*: assume "0 <= i2 & i2 <= sa..csize" */
5947        sa.add_at(i2, v2);
5948
5949        sb.add_at(i2, v2);
5950        sb.set(i1, v1);
5951
5952        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
5953    }
5954
5955    static void set_add_at_post_s_191(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2, Object v2)
5956    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
```

```
5957                 sa..contents = sb..contents & sa..csize = sb..csize"
5958         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                 "sb..msize"
5959         ensures "True" */
5960     {
5961         /*: assume "0 <= i1 & i1 < sa..csize" */
5962         sa.set(i1, v1);
5963         /*: assume "0 <= i2 & i2 <= sa..csize" */
5964         sa.add_at(i2, v2);
5965         /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & (i1, v2) :
                 sa..(old contents) & v1 = v2 & 0 <= i1 & i1 < sa..(old csize)) | (i1 > i2 &
                 (ALL v. ((i1, v) : sa..contents) = ((i1, v) : sa..(old contents)) & (i1, v1)
                 : sa..contents & (i1, v1) : sa..(old contents) & 0 <= i1 & i1 < sa..csize &
                 0 <= i1 & i1 < sa..(old csize)))" */
5966
5967         sb.add_at(i2, v2);
5968         sb.set(i1, v1);
5969
5970         /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
5971     }
5972
5973     static void set_get_pre_s_192(ArrayList sa, ArrayList sb, int i1, Object v1, int i2)
5974     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5975                 sa..contents = sb..contents & sa..csize = sb..csize"
5976         modifies "sa..contents", "sb..contents"
5977         ensures "True" */
5978     {
5979         /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & 0 <= i1 & i1 <
                 sa..csize) | i1 > i2" */
5980         /*: assume "0 <= i1 & i1 < sa..csize" */
5981         sa.set(i1, v1);
5982         /*: assume "0 <= i2 & i2 < sa..csize" */
5983         Object r2a = sa.get(i2);
5984
5985         Object r2b = sb.get(i2);
5986         sb.set(i1, v1);
5987
5988         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
5989     }
5990
5991     static void set_get_pre_c_192(ArrayList sa, ArrayList sb, int i1, Object v1, int i2)
5992     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5993                 sa..contents = sb..contents & sa..csize = sb..csize"
5994         modifies "sa..contents", "sb..contents"
5995         ensures "True" */
5996     {
5997         /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & 0 <= i1 & i1 <
                 sa..csize) | i1 > i2)" */
5998         /*: assume "0 <= i1 & i1 < sa..csize" */
5999         sa.set(i1, v1);
6000         /*: assume "0 <= i2 & i2 < sa..csize" */
6001         Object r2a = sa.get(i2);
6002
6003         /*: assume "0 <= i2 & i2 < sb..csize" */
6004         Object r2b = sb.get(i2);
6005         /*: assume "0 <= i1 & i1 < sb..csize" */
6006         sb.set(i1, v1);
6007
6008         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                 */
6009     }
6010
6011     static void set_get_between_s_193(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2)
6012     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
```

```
6013                 sa..contents = sb..contents & sa..csize = sb..csize"
6014         modifies "sa..contents", "sb..contents"
6015         ensures "True" */
6016     {
6017         /*: assume "0 <= i1 & i1 < sa..csize" */
6018         sa.set(i1, v1);
6019         /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & 0 <= i1 & i1 <
                sa..(old csize)) | i1 > i2" */
6020         /*: assume "0 <= i2 & i2 < sa..csize" */
6021         Object r2a = sa.get(i2);
6022 
6023         Object r2b = sb.get(i2);
6024         sb.set(i1, v1);
6025 
6026         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
6027     }
6028 
6029     static void set_get_between_c_193(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2)
6030     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6031                 sa..contents = sb..contents & sa..csize = sb..csize"
6032         modifies "sa..contents", "sb..contents"
6033         ensures "True" */
6034     {
6035         /*: assume "0 <= i1 & i1 < sa..csize" */
6036         sa.set(i1, v1);
6037         /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & 0 <= i1 & i1
                < sa..(old csize)) | i1 > i2)" */
6038         /*: assume "0 <= i2 & i2 < sa..csize" */
6039         Object r2a = sa.get(i2);
6040 
6041         /*: assume "0 <= i2 & i2 < sb..csize" */
6042         Object r2b = sb.get(i2);
6043         /*: assume "0 <= i1 & i1 < sb..csize" */
6044         sb.set(i1, v1);
6045 
6046         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
6047     }
6048 
6049     static void set_get_post_s_194(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2)
6050     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6051                 sa..contents = sb..contents & sa..csize = sb..csize"
6052         modifies "sa..contents", "sb..contents"
6053         ensures "True" */
6054     {
6055         /*: assume "0 <= i1 & i1 < sa..csize" */
6056         sa.set(i1, v1);
6057         /*: assume "0 <= i2 & i2 < sa..csize" */
6058         Object r2a = sa.get(i2);
6059         /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & 0 <= i1 & i1 <
                sa..(old csize)) | i1 > i2" */
6060 
6061         Object r2b = sb.get(i2);
6062         sb.set(i1, v1);
6063 
6064         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
6065     }
6066 
6067     static void set_get_post_c_194(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2)
6068     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6069                 sa..contents = sb..contents & sa..csize = sb..csize"
6070         modifies "sa..contents", "sb..contents"
```

```
6071        ensures "True" */
6072    {
6073        /*: assume "0 <= i1 & i1 < sa..csize" */
6074        sa.set(i1, v1);
6075        /*: assume "0 <= i2 & i2 < sa..csize" */
6076        Object r2a = sa.get(i2);
6077        /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & 0 <= i1 & i1
                < sa..(old csize)) | i1 > i2)" */
6078
6079        /*: assume "0 <= i2 & i2 < sb..csize" */
6080        Object r2b = sb.get(i2);
6081        /*: assume "0 <= i1 & i1 < sb..csize" */
6082        sb.set(i1, v1);
6083
6084        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
6085    }
6086
6087    static void set_indexOf_pre_s_195(ArrayList sa, ArrayList sb, int i1, Object v1,
            Object v2)
6088    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6089                  sa..contents = sb..contents & sa..csize = sb..csize"
6090        modifies "sa..contents", "sb..contents"
6091        ensures "True" */
6092    {
6093        /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
                v2) | (EX i. (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) :
                sa..contents & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 <
                sa..csize & v1 = v2) | (~(EX i. (i, v2) : sa..contents & 0 <= i & i <= i1) &
                (EX i. (i, v2) : sa..contents & i1 < i & i < sa..csize) & v1 ~= v2)" */
6094        /*: assume "0 <= i1 & i1 < sa..csize" */
6095        sa.set(i1, v1);
6096        int r2a = sa.indexOf(v2);
6097
6098        int r2b = sb.indexOf(v2);
6099        sb.set(i1, v1);
6100
6101        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
6102    }
6103
6104    static void set_indexOf_pre_c_195(ArrayList sa, ArrayList sb, int i1, Object v1,
            Object v2)
6105    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6106                  sa..contents = sb..contents & sa..csize = sb..csize"
6107        modifies "sa..contents", "sb..contents"
6108        ensures "True" */
6109    {
6110        /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
                v2) | (EX i. (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) :
                sa..contents & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 <
                sa..csize & v1 = v2) | (~(EX i. (i, v2) : sa..contents & 0 <= i & i <= i1) &
                (EX i. (i, v2) : sa..contents & i1 < i & i < sa..csize) & v1 ~= v2))" */
6111        /*: assume "0 <= i1 & i1 < sa..csize" */
6112        sa.set(i1, v1);
6113        int r2a = sa.indexOf(v2);
6114
6115        int r2b = sb.indexOf(v2);
6116        /*: assume "0 <= i1 & i1 < sb..csize" */
6117        sb.set(i1, v1);
6118
6119        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
6120    }
6121
```

```
6122    static void set_indexOf_between_s_196(ArrayList sa, ArrayList sb, int i1, Object
            v1, Object v2)
6123    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6124                 sa..contents = sb..contents & sa..csize = sb..csize"
6125        modifies "sa..contents", "sb..contents"
6126        ensures "True" */
6127    {
6128        /*: assume "0 <= i1 & i1 < sa..csize" */
6129        sa.set(i1, v1);
6130        /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & (i1,
                v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | (EX i. (i, v2)
                : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) : sa..contents & 0 <= i
                & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & (i1, v2) :
                sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | (~(EX i. (i, v2) :
                sa..contents & 0 <= i & i <= i1) & (EX i. (i, v2) : sa..contents & i1 < i &
                i < sa..csize) & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old
                csize))" */
6131        int r2a = sa.indexOf(v2);
6132
6133        int r2b = sb.indexOf(v2);
6134        sb.set(i1, v1);
6135
6136        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
6137    }
6138
6139    static void set_indexOf_between_c_196(ArrayList sa, ArrayList sb, int i1, Object
            v1, Object v2)
6140    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6141                 sa..contents = sb..contents & sa..csize = sb..csize"
6142        modifies "sa..contents", "sb..contents"
6143        ensures "True" */
6144    {
6145        /*: assume "0 <= i1 & i1 < sa..csize" */
6146        sa.set(i1, v1);
6147        /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & (i1,
                v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | (EX i. (i, v2)
                : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) : sa..contents & 0 <= i
                & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & (i1, v2) :
                sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | (~(EX i. (i, v2) :
                sa..contents & 0 <= i & i <= i1) & (EX i. (i, v2) : sa..contents & i1 < i &
                i < sa..csize) & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old
                csize)))" */
6148        int r2a = sa.indexOf(v2);
6149
6150        int r2b = sb.indexOf(v2);
6151        /*: assume "0 <= i1 & i1 < sb..csize" */
6152        sb.set(i1, v1);
6153
6154        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
6155    }
6156
6157    static void set_indexOf_post_s_197(ArrayList sa, ArrayList sb, int i1, Object v1,
            Object v2)
6158    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6159                 sa..contents = sb..contents & sa..csize = sb..csize"
6160        modifies "sa..contents", "sb..contents"
6161        ensures "True" */
6162    {
6163        /*: assume "0 <= i1 & i1 < sa..csize" */
6164        sa.set(i1, v1);
6165        int r2a = sa.indexOf(v2);
6166        /*: assume "(r2a < 0 & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old
                csize)) | (0 <= r2a & r2a < i1) | (r2a = i1 & (i1, v2) : sa..(old contents)
```

```
        & 0 <= i1 & i1 < sa..(old csize)) | (r2a > i1 & (i1, v2) ~: sa..(old
        contents) & 0 <= i1 & i1 < sa..(old csize))" */

    int r2b = sb.indexOf(v2);
    sb.set(i1, v1);

    /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
}

static void set_indexOf_post_c_197(ArrayList sa, ArrayList sb, int i1, Object v1,
    Object v2)
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
              sa..contents = sb..contents & sa..csize = sb..csize"
    modifies "sa..contents", "sb..contents"
    ensures "True" */
{
    /*: assume "0 <= i1 & i1 < sa..csize" */
    sa.set(i1, v1);
    int r2a = sa.indexOf(v2);
    /*: assume "~((r2a < 0 & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 <
        sa..(old csize)) | (0 <= r2a & r2a < i1) | (r2a = i1 & (i1, v2) : sa..(old
        contents) & 0 <= i1 & i1 < sa..(old csize)) | (r2a > i1 & (i1, v2) ~:
        sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)))" */

    int r2b = sb.indexOf(v2);
    /*: assume "0 <= i1 & i1 < sb..csize" */
    sb.set(i1, v1);

    /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
}

static void set_lastIndexOf_pre_s_198(ArrayList sa, ArrayList sb, int i1, Object
    v1, Object v2)
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
              sa..contents = sb..contents & sa..csize = sb..csize"
    modifies "sa..contents", "sb..contents"
    ensures "True" */
{
    /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
        v2) | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
        sa..contents & i1 <= i & i < sa..csize) & v1 ~= v2) | ((i1, v2) :
        sa..contents & 0 <= i1 & i1 < sa..csize & ~(EX i. (i, v2) : sa..contents &
        i1 < i & i < sa..csize) & v1 = v2) | (EX i. (i, v2) : sa..contents & i1 < i
        & i < sa..csize)" */
    /*: assume "0 <= i1 & i1 < sa..csize" */
    sa.set(i1, v1);
    int r2a = sa.lastIndexOf(v2);

    int r2b = sb.lastIndexOf(v2);
    sb.set(i1, v1);

    /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
}

static void set_lastIndexOf_pre_c_198(ArrayList sa, ArrayList sb, int i1, Object
    v1, Object v2)
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
              sa..contents = sb..contents & sa..csize = sb..csize"
    modifies "sa..contents", "sb..contents"
    ensures "True" */
{
    /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
        v2) | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
        sa..contents & i1 <= i & i < sa..csize) & v1 ~= v2) | ((i1, v2) :
```

```
            sa..contents & 0 <= i1 & i1 < sa..csize & ~(EX i. (i, v2) : sa..contents &
            i1 < i & i < sa..csize) & v1 = v2) | (EX i. (i, v2) : sa..contents & i1 < i
            & i < sa..csize))" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.set(i1, v1);
        int r2a = sa.lastIndexOf(v2);

        int r2b = sb.lastIndexOf(v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.set(i1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void set_lastIndexOf_between_s_199(ArrayList sa, ArrayList sb, int i1,
        Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.set(i1, v1);
        /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & (i1,
            v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | ((EX i. (i,
            v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) : sa..contents & i1
            <= i & i < sa..csize) & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 <
            sa..(old csize)) | ((i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize &
            ~(EX i. (i, v2) : sa..contents & i1 < i & i < sa..csize) & (i1, v2) :
            sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | (EX i. (i, v2) :
            sa..contents & i1 < i & i < sa..csize)" */
        int r2a = sa.lastIndexOf(v2);

        int r2b = sb.lastIndexOf(v2);
        sb.set(i1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void set_lastIndexOf_between_c_199(ArrayList sa, ArrayList sb, int i1,
        Object v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.set(i1, v1);
        /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & (i1,
            v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | ((EX i. (i,
            v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) : sa..contents & i1
            <= i & i < sa..csize) & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 <
            sa..(old csize)) | ((i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize &
            ~(EX i. (i, v2) : sa..contents & i1 < i & i < sa..csize) & (i1, v2) :
            sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | (EX i. (i, v2) :
            sa..contents & i1 < i & i < sa..csize))" */
        int r2a = sa.lastIndexOf(v2);

        int r2b = sb.lastIndexOf(v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.set(i1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
```

```
    }

    static void set_lastIndexOf_post_s_200(ArrayList sa, ArrayList sb, int i1, Object
        v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.set(i1, v1);
        int r2a = sa.lastIndexOf(v2);
        /*: assume "(r2a < 0 & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old
            csize)) | (0 <= r2a & r2a < i1 & (i1, v2) ~: sa..(old contents) & 0 <= i1 &
            i1 < sa..(old csize)) | (r2a = i1 & (i1, v2) : sa..(old contents) & 0 <= i1
            & i1 < sa..(old csize)) | r2a > i1" */

        int r2b = sb.lastIndexOf(v2);
        sb.set(i1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void set_lastIndexOf_post_c_200(ArrayList sa, ArrayList sb, int i1, Object
        v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.set(i1, v1);
        int r2a = sa.lastIndexOf(v2);
        /*: assume "~((r2a < 0 & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 <
            sa..(old csize)) | (0 <= r2a & r2a < i1 & (i1, v2) ~: sa..(old contents) & 0
            <= i1 & i1 < sa..(old csize)) | (r2a = i1 & (i1, v2) : sa..(old contents) &
            0 <= i1 & i1 < sa..(old csize)) | r2a > i1)" */

        int r2b = sb.lastIndexOf(v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.set(i1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void set_remove_at_pre_s_201(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (ALL v. ((i1, v) :
            sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
            (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
            1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
            sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
            (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
            1 < sa..csize)" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);

```

```
6309        Object r2b = sb.remove_at(i2);
6310        sb.set(i1, v1);
6311
6312        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
6313    }
6314
6315    static void set_remove_at_between_s_202(ArrayList sa, ArrayList sb, int i1, Object
            v1, int i2)
6316    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6317                  sa..contents = sb..contents & sa..csize = sb..csize"
6318        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
6319        ensures "True" */
6320    {
6321        /*: assume "0 <= i1 & i1 < sa..csize" */
6322        sa.set(i1, v1);
6323        /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (ALL v. ((i1, v) :
                sa..(old contents)) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..(old
                contents) & (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) & 0
                <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v.
                ((i1, v) : sa..(old contents)) = ((i1 + 1, v) : sa..contents)) & (i1, v1) :
                sa..(old contents) & (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old
                csize) & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
6324        /*: assume "0 <= i2 & i2 < sa..csize" */
6325        Object r2a = sa.remove_at(i2);
6326
6327        Object r2b = sb.remove_at(i2);
6328        sb.set(i1, v1);
6329
6330        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
6331    }
6332
6333    static void set_remove_at_post_s_203(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2)
6334    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6335                  sa..contents = sb..contents & sa..csize = sb..csize"
6336        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
6337        ensures "True" */
6338    {
6339        /*: assume "0 <= i1 & i1 < sa..csize" */
6340        sa.set(i1, v1);
6341        /*: assume "0 <= i2 & i2 < sa..csize" */
6342        Object r2a = sa.remove_at(i2);
6343        /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize & (ALL v. ((i1, v) : sa..(old
                contents)) = ((i1, v) : sa..contents)) & (i1, v1) : sa..(old contents) &
                (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 & i1 <
                sa..csize) | (sa..csize > i1 & i1 > i2 & (ALL v. ((i1, v) : sa..(old
                contents)) = ((i1, v) : sa..contents)) & (i1, v1) : sa..(old contents) &
                (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 & i1 <
                sa..csize)" */
6344
6345        Object r2b = sb.remove_at(i2);
6346        sb.set(i1, v1);
6347
6348        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
6349    }
6350
6351    static void set_remove_at_pre_s_204(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2)
6352    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6353                  sa..contents = sb..contents & sa..csize = sb..csize"
6354        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
6355        ensures "True" */
6356    {
6357        /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, v1) :
                sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
```

```
            > i2 & (ALL v. ((i1, v) : sa..contents) = ((i1 + 1, v) : sa..contents)) &
            (i1, v1) : sa..contents & (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 <
            sa..csize & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
6358        /*: assume "0 <= i1 & i1 < sa..csize" */
6359        sa.set(i1, v1);
6360        /*: assume "0 <= i2 & i2 < sa..csize" */
6361        sa.remove_at(i2);
6362
6363        sb.remove_at(i2);
6364        sb.set(i1, v1);
6365
6366        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
6367    }
6368
6369    static void set_remove_at_between_s_205(ArrayList sa, ArrayList sb, int i1, Object
        v1, int i2)
6370    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6371                  sa..contents = sb..contents & sa..csize = sb..csize"
6372        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
6373        ensures "True" */
6374    {
6375        /*: assume "0 <= i1 & i1 < sa..csize" */
6376        sa.set(i1, v1);
6377        /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, v1) :
            sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
            > i2 & (ALL v. ((i1, v) : sa..(old contents)) = ((i1 + 1, v) :
            sa..contents)) & (i1, v1) : sa..(old contents) & (i1 + 1, v1) : sa..contents
            & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
6378        /*: assume "0 <= i2 & i2 < sa..csize" */
6379        sa.remove_at(i2);
6380
6381        sb.remove_at(i2);
6382        sb.set(i1, v1);
6383
6384        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
6385    }
6386
6387    static void set_remove_at_post_s_206(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2)
6388    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6389                  sa..contents = sb..contents & sa..csize = sb..csize"
6390        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
6391        ensures "True" */
6392    {
6393        /*: assume "0 <= i1 & i1 < sa..csize" */
6394        sa.set(i1, v1);
6395        /*: assume "0 <= i2 & i2 < sa..csize" */
6396        sa.remove_at(i2);
6397        /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize & (i1, v1) : sa..contents & 0
            <= i1 & i1 < sa..csize) | (sa..csize > i1 & i1 > i2 & (ALL v. ((i1, v) :
            sa..(old contents)) = ((i1, v) : sa..(contents))) & (i1, v1) : sa..(old
            contents) & (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) & 0 <=
            i1 & i1 < sa..csize)" */
6398
6399        sb.remove_at(i2);
6400        sb.set(i1, v1);
6401
6402        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
6403    }
6404
6405    static void set_set_pre_s_207(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
6406    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6407                  sa..contents = sb..contents & sa..csize = sb..csize"
6408        modifies "sa..contents", "sb..contents"
```

```
6409        ensures "True" */
6410    {
6411        /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
                sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | i1 > i2" */
6412        /*: assume "0 <= i1 & i1 < sa..csize" */
6413        sa.set(i1, v1);
6414        /*: assume "0 <= i2 & i2 < sa..csize" */
6415        Object r2a = sa.set(i2, v2);
6416
6417        Object r2b = sb.set(i2, v2);
6418        sb.set(i1, v1);
6419
6420        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
6421    }
6422
6423    static void set_set_pre_c_207(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2, Object v2)
6424    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6425                  sa..contents = sb..contents & sa..csize = sb..csize"
6426        modifies "sa..contents", "sb..contents"
6427        ensures "True" */
6428    {
6429        /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
                sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | i1 > i2)" */
6430        /*: assume "0 <= i1 & i1 < sa..csize" */
6431        sa.set(i1, v1);
6432        /*: assume "0 <= i2 & i2 < sa..csize" */
6433        Object r2a = sa.set(i2, v2);
6434
6435        /*: assume "0 <= i2 & i2 < sb..csize" */
6436        Object r2b = sb.set(i2, v2);
6437        /*: assume "0 <= i1 & i1 < sb..csize" */
6438        sb.set(i1, v1);
6439
6440        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
6441    }
6442
6443    static void set_set_between_s_208(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2, Object v2)
6444    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6445                  sa..contents = sb..contents & sa..csize = sb..csize"
6446        modifies "sa..contents", "sb..contents"
6447        ensures "True" */
6448    {
6449        /*: assume "0 <= i1 & i1 < sa..csize" */
6450        sa.set(i1, v1);
6451        /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & (i1, v2) :
                sa..(old contents) & v1 = v2 & 0 <= i1 & i1 < sa..(old csize)) | i1 > i2" */
6452        /*: assume "0 <= i2 & i2 < sa..csize" */
6453        Object r2a = sa.set(i2, v2);
6454
6455        Object r2b = sb.set(i2, v2);
6456        sb.set(i1, v1);
6457
6458        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
6459    }
6460
6461    static void set_set_between_c_208(ArrayList sa, ArrayList sb, int i1, Object v1,
            int i2, Object v2)
6462    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6463                  sa..contents = sb..contents & sa..csize = sb..csize"
6464        modifies "sa..contents", "sb..contents"
6465        ensures "True" */
6466    {
```

```
6467        /*: assume "0 <= i1 & i1 < sa..csize" */
6468        sa.set(i1, v1);
6469        /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & (i1, v2) :
                sa..(old contents) & v1 = v2 & 0 <= i1 & i1 < sa..(old csize)) | i1 > i2)" */
6470        /*: assume "0 <= i2 & i2 < sa..csize" */
6471        Object r2a = sa.set(i2, v2);
6472
6473        /*: assume "0 <= i2 & i2 < sb..csize" */
6474        Object r2b = sb.set(i2, v2);
6475        /*: assume "0 <= i1 & i1 < sb..csize" */
6476        sb.set(i1, v1);
6477
6478        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
6479    }
6480
6481    static void set_set_post_s_209(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2, Object v2)
6482    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6483                  sa..contents = sb..contents & sa..csize = sb..csize"
6484        modifies "sa..contents", "sb..contents"
6485        ensures "True" */
6486    {
6487        /*: assume "0 <= i1 & i1 < sa..csize" */
6488        sa.set(i1, v1);
6489        /*: assume "0 <= i2 & i2 < sa..csize" */
6490        Object r2a = sa.set(i2, v2);
6491        /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & (i1, v2) :
                sa..(old contents) & v1 = v2 & 0 <= i1 & i1 < sa..(old csize)) | i1 > i2" */
6492
6493        Object r2b = sb.set(i2, v2);
6494        sb.set(i1, v1);
6495
6496        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
6497    }
6498
6499    static void set_set_post_c_209(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2, Object v2)
6500    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6501                  sa..contents = sb..contents & sa..csize = sb..csize"
6502        modifies "sa..contents", "sb..contents"
6503        ensures "True" */
6504    {
6505        /*: assume "0 <= i1 & i1 < sa..csize" */
6506        sa.set(i1, v1);
6507        /*: assume "0 <= i2 & i2 < sa..csize" */
6508        Object r2a = sa.set(i2, v2);
6509        /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & (i1, v2) :
                sa..(old contents) & v1 = v2 & 0 <= i1 & i1 < sa..(old csize)) | i1 > i2)" */
6510
6511        /*: assume "0 <= i2 & i2 < sb..csize" */
6512        Object r2b = sb.set(i2, v2);
6513        /*: assume "0 <= i1 & i1 < sb..csize" */
6514        sb.set(i1, v1);
6515
6516        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
6517    }
6518
6519    static void set_set_pre_s_210(ArrayList sa, ArrayList sb, int i1, Object v1, int
            i2, Object v2)
6520    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6521                  sa..contents = sb..contents & sa..csize = sb..csize"
6522        modifies "sa..contents", "sb..contents"
6523        ensures "True" */
```

```
6524    {
6525        /*: assume "i1 < i2 | (i1 = i2 & v1 = v2) | i1 > i2" */
6526        /*: assume "0 <= i1 & i1 < sa..csize" */
6527        sa.set(i1, v1);
6528        /*: assume "0 <= i2 & i2 < sa..csize" */
6529        sa.set(i2, v2);
6530
6531        sb.set(i2, v2);
6532        sb.set(i1, v1);
6533
6534        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
6535    }
6536
6537    static void set_set_pre_c_210(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
6538    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6539                  sa..contents = sb..contents & sa..csize = sb..csize"
6540        modifies "sa..contents", "sb..contents"
6541        ensures "True" */
6542    {
6543        /*: assume "~(i1 < i2 | (i1 = i2 & v1 = v2) | i1 > i2)" */
6544        /*: assume "0 <= i1 & i1 < sa..csize" */
6545        sa.set(i1, v1);
6546        /*: assume "0 <= i2 & i2 < sa..csize" */
6547        sa.set(i2, v2);
6548
6549        /*: assume "0 <= i2 & i2 < sb..csize" */
6550        sb.set(i2, v2);
6551        /*: assume "0 <= i1 & i1 < sb..csize" */
6552        sb.set(i1, v1);
6553
6554        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
6555    }
6556
6557    static void set_set_between_s_211(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2, Object v2)
6558    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6559                  sa..contents = sb..contents & sa..csize = sb..csize"
6560        modifies "sa..contents", "sb..contents"
6561        ensures "True" */
6562    {
6563        /*: assume "0 <= i1 & i1 < sa..csize" */
6564        sa.set(i1, v1);
6565        /*: assume "i1 < i2 | (i1 = i2 & v1 = v2) | i1 > i2" */
6566        /*: assume "0 <= i2 & i2 < sa..csize" */
6567        sa.set(i2, v2);
6568
6569        sb.set(i2, v2);
6570        sb.set(i1, v1);
6571
6572        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
6573    }
6574
6575    static void set_set_between_c_211(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2, Object v2)
6576    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6577                  sa..contents = sb..contents & sa..csize = sb..csize"
6578        modifies "sa..contents", "sb..contents"
6579        ensures "True" */
6580    {
6581        /*: assume "0 <= i1 & i1 < sa..csize" */
6582        sa.set(i1, v1);
6583        /*: assume "~(i1 < i2 | (i1 = i2 & v1 = v2) | i1 > i2)" */
6584        /*: assume "0 <= i2 & i2 < sa..csize" */
6585        sa.set(i2, v2);
```

```
6586
6587        /*: assume "0 <= i2 & i2 < sb..csize" */
6588        sb.set(i2, v2);
6589        /*: assume "0 <= i1 & i1 < sb..csize" */
6590        sb.set(i1, v1);
6591
6592        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
6593    }
6594
6595    static void set_set_post_s_212(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
6596    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6597                sa..contents = sb..contents & sa..csize = sb..csize"
6598        modifies "sa..contents", "sb..contents"
6599        ensures "True" */
6600    {
6601        /*: assume "0 <= i1 & i1 < sa..csize" */
6602        sa.set(i1, v1);
6603        /*: assume "0 <= i2 & i2 < sa..csize" */
6604        sa.set(i2, v2);
6605        /*: assume "i1 < i2 | (i1 = i2 & v1 = v2) | i1 > i2" */
6606
6607        sb.set(i2, v2);
6608        sb.set(i1, v1);
6609
6610        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
6611    }
6612
6613    static void set_set_post_c_212(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
6614    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6615                sa..contents = sb..contents & sa..csize = sb..csize"
6616        modifies "sa..contents", "sb..contents"
6617        ensures "True" */
6618    {
6619        /*: assume "0 <= i1 & i1 < sa..csize" */
6620        sa.set(i1, v1);
6621        /*: assume "0 <= i2 & i2 < sa..csize" */
6622        sa.set(i2, v2);
6623        /*: assume "~(i1 < i2 | (i1 = i2 & v1 = v2) | i1 > i2)" */
6624
6625        /*: assume "0 <= i2 & i2 < sb..csize" */
6626        sb.set(i2, v2);
6627        /*: assume "0 <= i1 & i1 < sb..csize" */
6628        sb.set(i1, v1);
6629
6630        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
6631    }
6632
6633    static void set_size_pre_s_213(ArrayList sa, ArrayList sb, int i1, Object v1)
6634    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6635                sa..contents = sb..contents & sa..csize = sb..csize"
6636        modifies "sa..contents", "sb..contents"
6637        ensures "True" */
6638    {
6639        /*: assume "True" */
6640        /*: assume "0 <= i1 & i1 < sa..csize" */
6641        sa.set(i1, v1);
6642        int r2a = sa.size();
6643
6644        int r2b = sb.size();
6645        sb.set(i1, v1);
6646
6647        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
6648    }
```

```
6649
6650    static void set_size_pre_c_213(ArrayList sa, ArrayList sb, int i1, Object v1)
6651    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6652                sa..contents = sb..contents & sa..csize = sb..csize"
6653        modifies "sa..contents", "sb..contents"
6654        ensures "True" */
6655    {
6656        /*: assume "~(True)" */
6657        /*: assume "0 <= i1 & i1 < sa..csize" */
6658        sa.set(i1, v1);
6659        int r2a = sa.size();
6660
6661        int r2b = sb.size();
6662        /*: assume "0 <= i1 & i1 < sb..csize" */
6663        sb.set(i1, v1);
6664
6665        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
6666    }
6667
6668    static void set_size_between_s_214(ArrayList sa, ArrayList sb, int i1, Object v1)
6669    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6670                sa..contents = sb..contents & sa..csize = sb..csize"
6671        modifies "sa..contents", "sb..contents"
6672        ensures "True" */
6673    {
6674        /*: assume "0 <= i1 & i1 < sa..csize" */
6675        sa.set(i1, v1);
6676        /*: assume "True" */
6677        int r2a = sa.size();
6678
6679        int r2b = sb.size();
6680        sb.set(i1, v1);
6681
6682        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
6683    }
6684
6685    static void set_size_between_c_214(ArrayList sa, ArrayList sb, int i1, Object v1)
6686    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6687                sa..contents = sb..contents & sa..csize = sb..csize"
6688        modifies "sa..contents", "sb..contents"
6689        ensures "True" */
6690    {
6691        /*: assume "0 <= i1 & i1 < sa..csize" */
6692        sa.set(i1, v1);
6693        /*: assume "~(True)" */
6694        int r2a = sa.size();
6695
6696        int r2b = sb.size();
6697        /*: assume "0 <= i1 & i1 < sb..csize" */
6698        sb.set(i1, v1);
6699
6700        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
6701    }
6702
6703    static void set_size_post_s_215(ArrayList sa, ArrayList sb, int i1, Object v1)
6704    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6705                sa..contents = sb..contents & sa..csize = sb..csize"
6706        modifies "sa..contents", "sb..contents"
6707        ensures "True" */
6708    {
6709        /*: assume "0 <= i1 & i1 < sa..csize" */
6710        sa.set(i1, v1);
6711        int r2a = sa.size();
```

```
        /*: assume "True" */

        int r2b = sb.size();
        sb.set(i1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void set_size_post_c_215(ArrayList sa, ArrayList sb, int i1, Object v1)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.set(i1, v1);
        int r2a = sa.size();
        /*: assume "~(True)" */

        int r2b = sb.size();
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.set(i1, v1);

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void size_add_at_pre_s_216(ArrayList sa, ArrayList sb, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "False" */
        int r1a = sa.size();
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        sb.add_at(i2, v2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void size_add_at_pre_c_216(ArrayList sa, ArrayList sb, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "~(False)" */
        int r1a = sa.size();
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        /*: assume "0 <= i2 & i2 <= sb..csize" */
        sb.add_at(i2, v2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }
```

```
    static void size_add_at_between_s_217(ArrayList sa, ArrayList sb, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "False" */
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        sb.add_at(i2, v2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void size_add_at_between_c_217(ArrayList sa, ArrayList sb, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "~(False)" */
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        /*: assume "0 <= i2 & i2 <= sb..csize" */
        sb.add_at(i2, v2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void size_add_at_post_s_218(ArrayList sa, ArrayList sb, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);
        /*: assume "False" */

        sb.add_at(i2, v2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void size_add_at_post_c_218(ArrayList sa, ArrayList sb, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "0 <= i2 & i2 <= sa..csize" */
```

```
6833        sa.add_at(i2, v2);
6834        /*: assume "~(False)" */
6835
6836        /*: assume "0 <= i2 & i2 <= sb..csize" */
6837        sb.add_at(i2, v2);
6838        int r1b = sb.size();
6839
6840        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
6841    }
6842
6843    static void size_get_pre_s_219(ArrayList sa, ArrayList sb, int i2)
6844    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6845                  sa..contents = sb..contents & sa..csize = sb..csize"
6846        ensures "True" */
6847    {
6848        /*: assume "True" */
6849        int r1a = sa.size();
6850        /*: assume "0 <= i2 & i2 < sa..csize" */
6851        Object r2a = sa.get(i2);
6852
6853        Object r2b = sb.get(i2);
6854        int r1b = sb.size();
6855
6856        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
6857    }
6858
6859    static void size_get_pre_c_219(ArrayList sa, ArrayList sb, int i2)
6860    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6861                  sa..contents = sb..contents & sa..csize = sb..csize"
6862        ensures "True" */
6863    {
6864        /*: assume "~(True)" */
6865        int r1a = sa.size();
6866        /*: assume "0 <= i2 & i2 < sa..csize" */
6867        Object r2a = sa.get(i2);
6868
6869        /*: assume "0 <= i2 & i2 < sb..csize" */
6870        Object r2b = sb.get(i2);
6871        int r1b = sb.size();
6872
6873        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
6874    }
6875
6876    static void size_get_between_s_220(ArrayList sa, ArrayList sb, int i2)
6877    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6878                  sa..contents = sb..contents & sa..csize = sb..csize"
6879        ensures "True" */
6880    {
6881        int r1a = sa.size();
6882        /*: assume "True" */
6883        /*: assume "0 <= i2 & i2 < sa..csize" */
6884        Object r2a = sa.get(i2);
6885
6886        Object r2b = sb.get(i2);
6887        int r1b = sb.size();
6888
6889        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
6890    }
6891
6892    static void size_get_between_c_220(ArrayList sa, ArrayList sb, int i2)
6893    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
```

```
6894                 sa..contents = sb..contents & sa..csize = sb..csize"
6895         ensures "True" */
6896     {
6897         int r1a = sa.size();
6898         /*: assume "~(True)" */
6899         /*: assume "0 <= i2 & i2 < sa..csize" */
6900         Object r2a = sa.get(i2);
6901 
6902         /*: assume "0 <= i2 & i2 < sb..csize" */
6903         Object r2b = sb.get(i2);
6904         int r1b = sb.size();
6905 
6906         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
6907     }
6908 
6909     static void size_get_post_s_221(ArrayList sa, ArrayList sb, int i2)
6910     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6911                 sa..contents = sb..contents & sa..csize = sb..csize"
6912         ensures "True" */
6913     {
6914         int r1a = sa.size();
6915         /*: assume "0 <= i2 & i2 < sa..csize" */
6916         Object r2a = sa.get(i2);
6917         /*: assume "True" */
6918 
6919         Object r2b = sb.get(i2);
6920         int r1b = sb.size();
6921 
6922         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
6923     }
6924 
6925     static void size_get_post_c_221(ArrayList sa, ArrayList sb, int i2)
6926     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6927                 sa..contents = sb..contents & sa..csize = sb..csize"
6928         ensures "True" */
6929     {
6930         int r1a = sa.size();
6931         /*: assume "0 <= i2 & i2 < sa..csize" */
6932         Object r2a = sa.get(i2);
6933         /*: assume "~(True)" */
6934 
6935         /*: assume "0 <= i2 & i2 < sb..csize" */
6936         Object r2b = sb.get(i2);
6937         int r1b = sb.size();
6938 
6939         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
6940     }
6941 
6942     static void size_indexOf_pre_s_222(ArrayList sa, ArrayList sb, Object v2)
6943     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6944                 sa..contents = sb..contents & sa..csize = sb..csize"
6945         ensures "True" */
6946     {
6947         /*: assume "True" */
6948         int r1a = sa.size();
6949         int r2a = sa.indexOf(v2);
6950 
6951         int r2b = sb.indexOf(v2);
6952         int r1b = sb.size();
6953 
6954         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
```

```
    }

    static void size_indexOf_pre_c_222(ArrayList sa, ArrayList sb, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "~(True)" */
        int r1a = sa.size();
        int r2a = sa.indexOf(v2);

        int r2b = sb.indexOf(v2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void size_indexOf_between_s_223(ArrayList sa, ArrayList sb, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "True" */
        int r2a = sa.indexOf(v2);

        int r2b = sb.indexOf(v2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void size_indexOf_between_c_223(ArrayList sa, ArrayList sb, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "~(True)" */
        int r2a = sa.indexOf(v2);

        int r2b = sb.indexOf(v2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void size_indexOf_post_s_224(ArrayList sa, ArrayList sb, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.size();
        int r2a = sa.indexOf(v2);
        /*: assume "True" */

        int r2b = sb.indexOf(v2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }
```

```
7016
7017    static void size_indexOf_post_c_224(ArrayList sa, ArrayList sb, Object v2)
7018    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7019                sa..contents = sb..contents & sa..csize = sb..csize"
7020        ensures "True" */
7021    {
7022        int r1a = sa.size();
7023        int r2a = sa.indexOf(v2);
7024        /*: assume "~(True)" */
7025
7026        int r2b = sb.indexOf(v2);
7027        int r1b = sb.size();
7028
7029        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
7030    }
7031
7032    static void size_lastIndexOf_pre_s_225(ArrayList sa, ArrayList sb, Object v2)
7033    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7034                sa..contents = sb..contents & sa..csize = sb..csize"
7035        ensures "True" */
7036    {
7037        /*: assume "True" */
7038        int r1a = sa.size();
7039        int r2a = sa.lastIndexOf(v2);
7040
7041        int r2b = sb.lastIndexOf(v2);
7042        int r1b = sb.size();
7043
7044        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
7045    }
7046
7047    static void size_lastIndexOf_pre_c_225(ArrayList sa, ArrayList sb, Object v2)
7048    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7049                sa..contents = sb..contents & sa..csize = sb..csize"
7050        ensures "True" */
7051    {
7052        /*: assume "~(True)" */
7053        int r1a = sa.size();
7054        int r2a = sa.lastIndexOf(v2);
7055
7056        int r2b = sb.lastIndexOf(v2);
7057        int r1b = sb.size();
7058
7059        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
7060    }
7061
7062    static void size_lastIndexOf_between_s_226(ArrayList sa, ArrayList sb, Object v2)
7063    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7064                sa..contents = sb..contents & sa..csize = sb..csize"
7065        ensures "True" */
7066    {
7067        int r1a = sa.size();
7068        /*: assume "True" */
7069        int r2a = sa.lastIndexOf(v2);
7070
7071        int r2b = sb.lastIndexOf(v2);
7072        int r1b = sb.size();
7073
7074        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
7075    }
7076
```

```
7077    static void size_lastIndexOf_between_c_226(ArrayList sa, ArrayList sb, Object v2)
7078    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7079                  sa..contents = sb..contents & sa..csize = sb..csize"
7080        ensures "True" */
7081    {
7082        int r1a = sa.size();
7083        /*: assume "~(True)" */
7084        int r2a = sa.lastIndexOf(v2);
7085
7086        int r2b = sb.lastIndexOf(v2);
7087        int r1b = sb.size();
7088
7089        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
7090    }
7091
7092    static void size_lastIndexOf_post_s_227(ArrayList sa, ArrayList sb, Object v2)
7093    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7094                  sa..contents = sb..contents & sa..csize = sb..csize"
7095        ensures "True" */
7096    {
7097        int r1a = sa.size();
7098        int r2a = sa.lastIndexOf(v2);
7099        /*: assume "True" */
7100
7101        int r2b = sb.lastIndexOf(v2);
7102        int r1b = sb.size();
7103
7104        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
7105    }
7106
7107    static void size_lastIndexOf_post_c_227(ArrayList sa, ArrayList sb, Object v2)
7108    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7109                  sa..contents = sb..contents & sa..csize = sb..csize"
7110        ensures "True" */
7111    {
7112        int r1a = sa.size();
7113        int r2a = sa.lastIndexOf(v2);
7114        /*: assume "~(True)" */
7115
7116        int r2b = sb.lastIndexOf(v2);
7117        int r1b = sb.size();
7118
7119        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
7120    }
7121
7122    static void size_remove_at_pre_s_228(ArrayList sa, ArrayList sb, int i2)
7123    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7124                  sa..contents = sb..contents & sa..csize = sb..csize"
7125        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7126        ensures "True" */
7127    {
7128        /*: assume "False" */
7129        int r1a = sa.size();
7130        /*: assume "0 <= i2 & i2 < sa..csize" */
7131        Object r2a = sa.remove_at(i2);
7132
7133        Object r2b = sb.remove_at(i2);
7134        int r1b = sb.size();
7135
7136        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
7137    }
```

```
7138
7139    static void size_remove_at_pre_c_228(ArrayList sa, ArrayList sb, int i2)
7140    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7141                sa..contents = sb..contents & sa..csize = sb..csize"
7142        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7143        ensures "True" */
7144    {
7145        /*: assume "~(False)" */
7146        int r1a = sa.size();
7147        /*: assume "0 <= i2 & i2 < sa..csize" */
7148        Object r2a = sa.remove_at(i2);
7149
7150        /*: assume "0 <= i2 & i2 < sb..csize" */
7151        Object r2b = sb.remove_at(i2);
7152        int r1b = sb.size();
7153
7154        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
7155    }
7156
7157    static void size_remove_at_between_s_229(ArrayList sa, ArrayList sb, int i2)
7158    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7159                sa..contents = sb..contents & sa..csize = sb..csize"
7160        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7161        ensures "True" */
7162    {
7163        int r1a = sa.size();
7164        /*: assume "False" */
7165        /*: assume "0 <= i2 & i2 < sa..csize" */
7166        Object r2a = sa.remove_at(i2);
7167
7168        Object r2b = sb.remove_at(i2);
7169        int r1b = sb.size();
7170
7171        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
7172    }
7173
7174    static void size_remove_at_between_c_229(ArrayList sa, ArrayList sb, int i2)
7175    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7176                sa..contents = sb..contents & sa..csize = sb..csize"
7177        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7178        ensures "True" */
7179    {
7180        int r1a = sa.size();
7181        /*: assume "~(False)" */
7182        /*: assume "0 <= i2 & i2 < sa..csize" */
7183        Object r2a = sa.remove_at(i2);
7184
7185        /*: assume "0 <= i2 & i2 < sb..csize" */
7186        Object r2b = sb.remove_at(i2);
7187        int r1b = sb.size();
7188
7189        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
7190    }
7191
7192    static void size_remove_at_post_s_230(ArrayList sa, ArrayList sb, int i2)
7193    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7194                sa..contents = sb..contents & sa..csize = sb..csize"
7195        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7196        ensures "True" */
7197    {
7198        int r1a = sa.size();
7199        /*: assume "0 <= i2 & i2 < sa..csize" */
```

```
7200        Object r2a = sa.remove_at(i2);
7201        /*: assume "False" */
7202
7203        Object r2b = sb.remove_at(i2);
7204        int r1b = sb.size();
7205
7206        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
7207    }
7208
7209    static void size_remove_at_post_c_230(ArrayList sa, ArrayList sb, int i2)
7210    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7211                  sa..contents = sb..contents & sa..csize = sb..csize"
7212        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7213        ensures "True" */
7214    {
7215        int r1a = sa.size();
7216        /*: assume "0 <= i2 & i2 < sa..csize" */
7217        Object r2a = sa.remove_at(i2);
7218        /*: assume "~(False)" */
7219
7220        /*: assume "0 <= i2 & i2 < sb..csize" */
7221        Object r2b = sb.remove_at(i2);
7222        int r1b = sb.size();
7223
7224        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
7225    }
7226
7227    static void size_remove_at_pre_s_231(ArrayList sa, ArrayList sb, int i2)
7228    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7229                  sa..contents = sb..contents & sa..csize = sb..csize"
7230        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7231        ensures "True" */
7232    {
7233        /*: assume "False" */
7234        int r1a = sa.size();
7235        /*: assume "0 <= i2 & i2 < sa..csize" */
7236        sa.remove_at(i2);
7237
7238        sb.remove_at(i2);
7239        int r1b = sb.size();
7240
7241        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
7242    }
7243
7244    static void size_remove_at_pre_c_231(ArrayList sa, ArrayList sb, int i2)
7245    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7246                  sa..contents = sb..contents & sa..csize = sb..csize"
7247        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7248        ensures "True" */
7249    {
7250        /*: assume "~(False)" */
7251        int r1a = sa.size();
7252        /*: assume "0 <= i2 & i2 < sa..csize" */
7253        sa.remove_at(i2);
7254
7255        /*: assume "0 <= i2 & i2 < sb..csize" */
7256        sb.remove_at(i2);
7257        int r1b = sb.size();
7258
7259        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
7260    }
7261
```

```
    static void size_remove_at_between_s_232(ArrayList sa, ArrayList sb, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "False" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        sb.remove_at(i2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void size_remove_at_between_c_232(ArrayList sa, ArrayList sb, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "~(False)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void size_remove_at_post_s_233(ArrayList sa, ArrayList sb, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);
        /*: assume "False" */

        sb.remove_at(i2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void size_remove_at_post_c_233(ArrayList sa, ArrayList sb, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);
        /*: assume "~(False)" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
```

```
        sb.remove_at(i2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)" */
    }

    static void size_set_pre_s_234(ArrayList sa, ArrayList sb, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "True" */
        int r1a = sa.size();
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        Object r2b = sb.set(i2, v2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void size_set_pre_c_234(ArrayList sa, ArrayList sb, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "~(True)" */
        int r1a = sa.size();
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.set(i2, v2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)" */
    }

    static void size_set_between_s_235(ArrayList sa, ArrayList sb, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "True" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.set(i2, v2);

        Object r2b = sb.set(i2, v2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void size_set_between_c_235(ArrayList sa, ArrayList sb, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
```

```
7387        modifies "sa..contents", "sb..contents"
7388        ensures "True" */
7389    {
7390        int r1a = sa.size();
7391        /*: assume "~(True)" */
7392        /*: assume "0 <= i2 & i2 < sa..csize" */
7393        Object r2a = sa.set(i2, v2);
7394
7395        /*: assume "0 <= i2 & i2 < sb..csize" */
7396        Object r2b = sb.set(i2, v2);
7397        int r1b = sb.size();
7398
7399        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
7400    }
7401
7402    static void size_set_post_s_236(ArrayList sa, ArrayList sb, int i2, Object v2)
7403    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7404                  sa..contents = sb..contents & sa..csize = sb..csize"
7405        modifies "sa..contents", "sb..contents"
7406        ensures "True" */
7407    {
7408        int r1a = sa.size();
7409        /*: assume "0 <= i2 & i2 < sa..csize" */
7410        Object r2a = sa.set(i2, v2);
7411        /*: assume "True" */
7412
7413        Object r2b = sb.set(i2, v2);
7414        int r1b = sb.size();
7415
7416        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
7417    }
7418
7419    static void size_set_post_c_236(ArrayList sa, ArrayList sb, int i2, Object v2)
7420    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7421                  sa..contents = sb..contents & sa..csize = sb..csize"
7422        modifies "sa..contents", "sb..contents"
7423        ensures "True" */
7424    {
7425        int r1a = sa.size();
7426        /*: assume "0 <= i2 & i2 < sa..csize" */
7427        Object r2a = sa.set(i2, v2);
7428        /*: assume "~(True)" */
7429
7430        /*: assume "0 <= i2 & i2 < sb..csize" */
7431        Object r2b = sb.set(i2, v2);
7432        int r1b = sb.size();
7433
7434        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
7435    }
7436
7437    static void size_set_pre_s_237(ArrayList sa, ArrayList sb, int i2, Object v2)
7438    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7439                  sa..contents = sb..contents & sa..csize = sb..csize"
7440        modifies "sa..contents", "sb..contents"
7441        ensures "True" */
7442    {
7443        /*: assume "True" */
7444        int r1a = sa.size();
7445        /*: assume "0 <= i2 & i2 < sa..csize" */
7446        sa.set(i2, v2);
7447
7448        sb.set(i2, v2);
```

```
        int r1b = sb.size();

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void size_set_pre_c_237(ArrayList sa, ArrayList sb, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        /*: assume "~(True)" */
        int r1a = sa.size();
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.set(i2, v2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void size_set_between_s_238(ArrayList sa, ArrayList sb, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "True" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        sb.set(i2, v2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void size_set_between_c_238(ArrayList sa, ArrayList sb, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "~(True)" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.set(i2, v2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void size_set_post_s_239(ArrayList sa, ArrayList sb, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
```

```
    {
        int r1a = sa.size();
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);
        /*: assume "True" */

        sb.set(i2, v2);
        int r1b = sb.size();

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void size_set_post_c_239(ArrayList sa, ArrayList sb, int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);
        /*: assume "~(True)" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.set(i2, v2);
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void size_size_pre_s_240(ArrayList sa, ArrayList sb)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "True" */
        int r1a = sa.size();
        int r2a = sa.size();

        int r2b = sb.size();
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void size_size_pre_c_240(ArrayList sa, ArrayList sb)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        /*: assume "~(True)" */
        int r1a = sa.size();
        int r2a = sa.size();

        int r2b = sb.size();
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void size_size_between_s_241(ArrayList sa, ArrayList sb)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
```

```
                    sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "True" */
        int r2a = sa.size();

        int r2b = sb.size();
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void size_size_between_c_241(ArrayList sa, ArrayList sb)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                    sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.size();
        /*: assume "~(True)" */
        int r2a = sa.size();

        int r2b = sb.size();
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void size_size_post_s_242(ArrayList sa, ArrayList sb)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                    sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.size();
        int r2a = sa.size();
        /*: assume "True" */

        int r2b = sb.size();
        int r1b = sb.size();

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void size_size_post_c_242(ArrayList sa, ArrayList sb)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                    sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
    {
        int r1a = sa.size();
        int r2a = sa.size();
        /*: assume "~(True)" */

        int r2b = sb.size();
        int r1b = sb.size();

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

}
```

Listing 19. `ArrayListComm.java`

```
class ArrayListComm {
    static void add_at_indexOf_between_s_7(ArrayList sa, ArrayList sb, int i1, Object
        v1, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
            (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) : sa..contents
            & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize &
            (i1 + 1, v2) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
        {
            /*: assuming "(EX i. (i, v2) : sa..(old contents) & i1 <= i & i < sa..(old
                csize))" */
            {
                /*: pickWitness j :: int suchThat "(j, v2) : sa..(old contents) & i1 <=
                    j & j < sa..(old csize)" */
                /*: note "(j + 1, v2) : sa..contents & i1 + 1 <= j + 1 & j + 1 <
                    sa..csize" */
                /*: note "(EX i. (i, v2) : sa..contents & i1 + 1 <= i & i < sa..csize)"
                    */
            }
            /*: note "(EX i. (i, v2) : sa..contents & i1 + 1 <= i & i < sa..csize)" */
        }
        int r2a = sa.indexOf(v2);

        int r2b = sb.indexOf(v2);
        sb.add_at(i1, v1);

        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void add_at_indexOf_post_s_8(ArrayList sa, ArrayList sb, int i1, Object v1,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        {
            /*: assuming "(EX i. (i, v2) : sa..(old contents) & i1 <= i & i < sa..(old
                csize))" */
            {
                /*: pickWitness j :: int suchThat "(j, v2) : sa..(old contents) & i1 <=
                    j & j < sa..(old csize)" */
                /*: note "(j + 1, v2) : sa..contents & i1 + 1 <= j + 1 & j + 1 <
                    sa..csize" */
                /*: note "(EX i. (i, v2) : sa..contents & i1 + 1 <= i & i < sa..csize)"
                    */
            }
            /*: note "(EX i. (i, v2) : sa..contents & i1 + 1 <= i & i < sa..csize)" */
        }
        int r2a = sa.indexOf(v2);
        /*: assume "r2a < 0 | (0 <= r2a & r2a < i1) | (r2a = i1 & (i1 + 1, v2) :
            sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */

        int r2b = sb.indexOf(v2);
```

```
49          sb.add_at(i1, v1);
50
51          /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
52      }
53
54      static void add_at_lastIndexOf_pre_c_9(ArrayList sa, ArrayList sb, int i1, Object
            v1, Object v2)
55      /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
56                    sa..contents = sb..contents & sa..csize = sb..csize"
57          modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
58          ensures "True" */
59      {
60          /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
                v2) | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
                sa..contents & i1 <= i & i < sa..csize) & v1 ~= v2))" */
61          /*: assume "0 <= i1 & i1 <= sa..csize" */
62          sa.add_at(i1, v1);
63          /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
64          int r2a = sa.lastIndexOf(v2);
65
66          int r2b = sb.lastIndexOf(v2);
67          /*: assume "0 <= i1 & i1 <= sb..csize" */
68          sb.add_at(i1, v1);
69
70          {
71              /*: assuming "(EX i. (i, v2) : sa..(old contents) & i1 <= i & i < sa..(old
                    csize))" */
72              /*: note "(r2b + 1, v2) : sa__contents" */
73          }
74
75          /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
76      }
77
78      static void add_at_lastIndexOf_between_s_10(ArrayList sa, ArrayList sb, int i1,
            Object v1, Object v2)
79      /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
80                    sa..contents = sb..contents & sa..csize = sb..csize"
81          modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
82          ensures "True" */
83      {
84          /*: assume "0 <= i1 & i1 <= sa..csize" */
85          sa.add_at(i1, v1);
86          /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | ((EX i.
                (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) : sa..contents &
                i1 <= i & i < sa..csize))" */
87          {
88              /*: assuming "(EX i. (i, v2) : sa..(old contents) & i1 <= i & i < sa..(old
                    csize))" */
89              {
90                  /*: pickWitness j :: int suchThat "(j, v2) : sa..(old contents) & i1 <=
                        j & j < sa..(old csize)" */
91                  /*: note "(j + 1, v2) : sa..contents & i1 + 1 <= j + 1 & j + 1 <
                        sa..csize" */
92                  /*: note "(EX i. (i, v2) : sa..contents & i1 + 1 <= i & i < sa..csize)"
                        */
93              }
94              /*: note "(EX i. (i, v2) : sa..contents & i1 + 1 <= i & i < sa..csize)" */
95          }
96          int r2a = sa.lastIndexOf(v2);
97
98          int r2b = sb.lastIndexOf(v2);
99          sb.add_at(i1, v1);
```

```
100
101         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
102     }
103
104     static void add_at_lastIndexOf_between_c_10(ArrayList sa, ArrayList sb, int i1,
        Object v1, Object v2)
105     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
106                   sa..contents = sb..contents & sa..csize = sb..csize"
107         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
108         ensures "True" */
109     {
110         /*: assume "0 <= i1 & i1 <= sa..csize" */
111         sa.add_at(i1, v1);
112         /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
113         /*: assume "~(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | ((EX
                i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
                sa..contents & i1 <= i & i < sa..csize)))" */
114         int r2a = sa.lastIndexOf(v2);
115
116         int r2b = sb.lastIndexOf(v2);
117         /*: assume "0 <= i1 & i1 <= sb..csize" */
118         sb.add_at(i1, v1);
119
120         {
121             /*: assuming "(EX i. (i, v2) : sa..(old contents) & i1 <= i & i < sa..(old
                    csize))" */
122             /*: note "(r2b + 1, v2) : sa__contents" */
123         }
124
125         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
126     }
127
128     static void add_at_lastIndexOf_post_s_11(ArrayList sa, ArrayList sb, int i1, Object
        v1, Object v2)
129     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
130                   sa..contents = sb..contents & sa..csize = sb..csize"
131         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
132         ensures "True" */
133     {
134         /*: assume "0 <= i1 & i1 <= sa..csize" */
135         sa.add_at(i1, v1);
136         {
137             /*: assuming "(EX i. (i, v2) : sa..(old contents) & i1 <= i & i < sa..(old
                    csize))" */
138             {
139                 /*: pickWitness j :: int suchThat "(j, v2) : sa..(old contents) & i1 <=
                        j & j < sa..(old csize)" */
140                 /*: note "(j + 1, v2) : sa..contents & i1 + 1 <= j + 1 & j + 1 <
                        sa..csize" */
141                 /*: note "(EX i. (i, v2) : sa..contents & i1 + 1 <= i & i < sa..csize)"
                        */
142             }
143             /*: note "(EX i. (i, v2) : sa..contents & i1 + 1 <= i & i < sa..csize)" */
144         }
145         int r2a = sa.lastIndexOf(v2);
146         /*: assume "r2a < 0 | (0 <= r2a & r2a < i1)" */
147
148         int r2b = sb.lastIndexOf(v2);
149         sb.add_at(i1, v1);
150
151         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
152     }
```

```
153
154     static void add_at_lastIndexOf_post_c_11(ArrayList sa, ArrayList sb, int i1, Object
            v1, Object v2)
155     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
156                   sa..contents = sb..contents & sa..csize = sb..csize"
157         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
158         ensures "True" */
159     {
160         /*: assume "0 <= i1 & i1 <= sa..csize" */
161         sa.add_at(i1, v1);
162         /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
163         int r2a = sa.lastIndexOf(v2);
164         /*: assume "~(r2a < 0 | (0 <= r2a & r2a < i1))" */
165
166         int r2b = sb.lastIndexOf(v2);
167         /*: assume "0 <= i1 & i1 <= sb..csize" */
168         sb.add_at(i1, v1);
169
170         {
171             /*: assuming "(EX i. (i, v2) : sa..(old contents) & i1 <= i & i < sa..(old
                    csize))" */
172             /*: note "(r2b + 1, v2) : sa__contents" */
173         }
174
175         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
176     }
177
178     static void add_at_remove_at_pre_c_15(ArrayList sa, ArrayList sb, int i1, Object
            v1, int i2)
179     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
180                   sa..contents = sb..contents & sa..csize = sb..csize"
181         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
182         ensures "True" */
183     {
184         /*: assume "~((i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents)
                = ((i2, v) : sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 &
                i2 < sa..csize) | (sa..csize - 1 >= i1 & i1 = i2 & (i1, v1) : sa..contents &
                0 <= i1 & i1 < sa..csize) | (sa..csize - 1 >= i1 & i1 > i2 & (i1, v1) :
                sa..contents & 0 <= i1 & i1 < sa..csize))" */
185         /*: assume "0 <= i1 & i1 <= sa..csize" */
186         sa.add_at(i1, v1);
187         /*: assume "0 <= i2 & i2 < sa..csize" */
188         sa.remove_at(i2);
189
190         /*: assume "0 <= i2 & i2 < sb..csize" */
191         sb.remove_at(i2);
192         /*: assume "0 <= i1 & i1 <= sb..csize" */
193         sb.add_at(i1, v1);
194
195         {
196             /*: assuming "i1 < i2 & ~(ALL v. ((i2 - 1, v) : sa..(old contents)) = ((i2,
                    v) : sa..(old contents)))" */
197             {
198                 /*: pickWitness w :: obj suchThat "((i2 - 1, w) ~: sa..(old contents))
                        & ((i2, w) : sa..(old contents))" */
199                 /*: note "(i2, w) : sa..contents" */
200                 /*: note "(i2, w) ~: sb..contents" */
201                 /*: note "sa..contents ~= sb..contents" */
202             }
203             /*: note "sa..contents ~= sb..contents" */
204         }
205
```

```
        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
    }

    static void add_at_remove_at_between_c_16(ArrayList sa, ArrayList sb, int i1,
        Object v1, int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "~((i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa..contents)
            = ((i2 + 1, v) : sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 &
            i2 + 1 < sa..csize) | (sa..csize - 2 >= i1 & i1 = i2 & (i1 + 1, v1) :
            sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 2 >= i1 & i1
            > i2 & (i1 + 1, v1) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 <= sb..csize" */
        sb.add_at(i1, v1);

        {
            /*: assuming "i1 < i2 & ~(ALL v. ((i2 - 1, v) : sa..(old contents)) = ((i2,
                v) : sa..(old contents)))" */
            {
                /*: pickWitness w :: obj suchThat "((i2 - 1, w) ~: sa..(old contents))
                    & ((i2, w) : sa..(old contents))" */
                /*: note "(i2, w) : sa..contents" */
                /*: note "(i2, w) ~: sb..contents" */
                /*: note "sa..contents ~= sb..contents" */
            }
            /*: note "sa..contents ~= sb..contents" */
        }

        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
    }

    static void add_at_set_pre_c_21(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "~((i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents)
            = ((i2, v) : sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) :
            sa..contents & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize)
            | (i1 = i2 & i2 < sa..csize & (i2, v1) : sa..contents & (i2, v2) :
            sa..contents & v1 = v2 & 0 <= i2 & i2 < sa..csize) | i1 > i2)" */
        /*: assume "0 <= i1 & i1 <= sa..csize" */
        sa.add_at(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.set(i2, v2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.set(i2, v2);
        /*: assume "0 <= i1 & i1 <= sb..csize" */
        sb.add_at(i1, v1);

```

```
257         {
258             /*: assuming "i1 <= i2 & (i2, v2) ~: sa..(old contents)" */
259             /*: note "(i2 + 1, v2) ~: sa..contents" */
260             /*: note "(i2 + 1, v2) : sb..contents" */
261             /*: note "sa..contents ~= sb..contents" */
262         }
263 
264         /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
265     }
266 
267     static void indexOf_remove_at_pre_s_66(ArrayList sa, ArrayList sb, Object v1, int
        i2)
268     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
269                 sa..contents = sb..contents & sa..csize = sb..csize"
270         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
271         ensures "True" */
272     {
273         /*: assume "~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
                (i, v1) : sa..contents & 0 <= i & i < i2) | (~(EX i. (i, v1) : sa..contents
                & 0 <= i & i < i2) & (i2, v1) : sa..contents & 0 <= i2 & i2 < sa..csize & i2
                < sa..csize - 1 & (i2 + 1, v1) : sa..contents & 0 <= i2 + 1 & i2 + 1 <
                sa..csize)" */
274         int r1a = sa.indexOf(v1);
275         /*: assume "0 <= i2 & i2 < sa..csize" */
276         Object r2a = sa.remove_at(i2);
277 
278         Object r2b = sb.remove_at(i2);
279         /*: note "(~(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < i2) & (i2, v1) :
                sa..(old contents) & 0 <= i2 & i2 < sa..(old csize) & i2 < sa..(old csize) -
                1 & (i2 + 1, v1) : sa..(old contents) & 0 <= i2 + 1 & i2 + 1 < sa..(old
                csize)) --> (i2, v1) : sb..contents" */
280         int r1b = sb.indexOf(v1);
281 
282         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
283     }
284 
285     static void indexOf_remove_at_pre_c_66(ArrayList sa, ArrayList sb, Object v1, int
        i2)
286     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
287                 sa..contents = sb..contents & sa..csize = sb..csize"
288         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
289         ensures "True" */
290     {
291         /*: assume "~(~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
                (i, v1) : sa..contents & 0 <= i & i < i2) | (~(EX i. (i, v1) : sa..contents
                & 0 <= i & i < i2) & (i2, v1) : sa..contents & 0 <= i2 & i2 < sa..csize & i2
                < sa..csize - 1 & (i2 + 1, v1) : sa..contents & 0 <= i2 + 1 & i2 + 1 <
                sa..csize))" */
292         int r1a = sa.indexOf(v1);
293         /*: assume "0 <= i2 & i2 < sa..csize" */
294         Object r2a = sa.remove_at(i2);
295 
296         /*: assume "0 <= i2 & i2 < sb..csize" */
297         Object r2b = sb.remove_at(i2);
298         /*: ghost specvar sb__contents :: "(int * obj) set" = "sb..contents" */
299         int r1b = sb.indexOf(v1);
300 
301         {
302             /*: assuming "(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < sa..(old
                    csize)) & ~(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < i2) & (i2,
                    v1) ~: sa..(old contents)" */
303             /*: note "(r1a - 1, v1) : sb__contents" */
304         }
305 
```

```
306         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
307     }
308
309     static void indexOf_remove_at_between_s_67(ArrayList sa, ArrayList sb, Object v1,
            int i2)
310     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
311                   sa..contents = sb..contents & sa..csize = sb..csize"
312         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
313         ensures "True" */
314     {
315         int r1a = sa.indexOf(v1);
316         /*: assume "r1a < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize - 1 &
                (i2 + 1, v1) : sa..contents & 0 <= i2 + 1 & i2 + 1 < sa..csize)" */
317         /*: assume "0 <= i2 & i2 < sa..csize" */
318         Object r2a = sa.remove_at(i2);
319
320         Object r2b = sb.remove_at(i2);
321         /*: note "(~(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < i2) & (i2, v1) :
                sa..(old contents) & 0 <= i2 & i2 < sa..(old csize) & i2 < sa..(old csize) -
                1 & (i2 + 1, v1) : sa..(old contents) & 0 <= i2 + 1 & i2 + 1 < sa..(old
                csize)) --> (i2, v1) : sb..contents" */
322         int r1b = sb.indexOf(v1);
323
324         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
325     }
326
327     static void indexOf_remove_at_between_c_67(ArrayList sa, ArrayList sb, Object v1,
            int i2)
328     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
329                   sa..contents = sb..contents & sa..csize = sb..csize"
330         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
331         ensures "True" */
332     {
333         int r1a = sa.indexOf(v1);
334         /*: assume "~(r1a < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize - 1
                & (i2 + 1, v1) : sa..contents & 0 <= i2 + 1 & i2 + 1 < sa..csize))" */
335         /*: assume "0 <= i2 & i2 < sa..csize" */
336         Object r2a = sa.remove_at(i2);
337
338         /*: assume "0 <= i2 & i2 < sb..csize" */
339         Object r2b = sb.remove_at(i2);
340         /*: ghost specvar sb__contents :: "(int * obj) set" = "sb..contents" */
341         int r1b = sb.indexOf(v1);
342
343         {
344             /*: assuming "(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < sa..(old
                    csize)) & ~(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < i2) & (i2,
                    v1) ~: sa..(old contents)" */
345             /*: note "(r1a - 1, v1) : sb__contents" */
346         }
347
348         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
349     }
350
351     static void indexOf_remove_at_post_s_68(ArrayList sa, ArrayList sb, Object v1, int
            i2)
352     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
353                   sa..contents = sb..contents & sa..csize = sb..csize"
354         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
355         ensures "True" */
356     {
357         int r1a = sa.indexOf(v1);
```

```
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);
        /*: assume "r1a < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize & (i2,
            v1) : sa..contents & 0 <= i2 & i2 < sa..csize)" */

        Object r2b = sb.remove_at(i2);
        /*: note "(~(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < i2) & (i2, v1) :
            sa..(old contents) & 0 <= i2 & i2 < sa..(old csize) & i2 < sa..(old csize) -
            1 & (i2 + 1, v1) : sa..(old contents) & 0 <= i2 + 1 & i2 + 1 < sa..(old
            csize)) --> (i2, v1) : sb..contents" */
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void indexOf_remove_at_pre_s_69(ArrayList sa, ArrayList sb, Object v1, int
        i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
            (i, v1) : sa..contents & 0 <= i & i < i2) | (~(EX i. (i, v1) : sa..contents
            & 0 <= i & i < i2) & (i2, v1) : sa..contents & 0 <= i2 & i2 < sa..csize & i2
            < sa..csize - 1 & (i2 + 1, v1) : sa..contents & 0 <= i2 + 1 & i2 + 1 <
            sa..csize)" */
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        sb.remove_at(i2);
        /*: note "(~(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < i2) & (i2, v1) :
            sa..(old contents) & 0 <= i2 & i2 < sa..(old csize) & i2 < sa..(old csize) -
            1 & (i2 + 1, v1) : sa..(old contents) & 0 <= i2 + 1 & i2 + 1 < sa..(old
            csize)) --> (i2, v1) : sb..contents" */
        int r1b = sb.indexOf(v1);

        /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
    }

    static void indexOf_remove_at_pre_c_69(ArrayList sa, ArrayList sb, Object v1, int
        i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~(~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
            (i, v1) : sa..contents & 0 <= i & i < i2) | (~(EX i. (i, v1) : sa..contents
            & 0 <= i & i < i2) & (i2, v1) : sa..contents & 0 <= i2 & i2 < sa..csize & i2
            < sa..csize - 1 & (i2 + 1, v1) : sa..contents & 0 <= i2 + 1 & i2 + 1 <
            sa..csize))" */
        int r1a = sa.indexOf(v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        /*: ghost specvar sb__contents :: "(int * obj) set" = "sb..contents" */
        int r1b = sb.indexOf(v1);

        {
```

```
404             /*: assuming "(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < sa..(old
                    csize)) & ~(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < i2) & (i2,
                    v1) ~: sa..(old contents)" */
405             /*: note "(r1a - 1, v1) : sb__contents" */
406         }
407
408         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
409     }
410
411     static void indexOf_remove_at_between_s_70(ArrayList sa, ArrayList sb, Object v1,
            int i2)
412     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
413                     sa..contents = sb..contents & sa..csize = sb..csize"
414         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
415         ensures "True" */
416     {
417         int r1a = sa.indexOf(v1);
418         /*: assume "r1a < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize - 1 &
                (i2 + 1, v1) : sa..contents & 0 <= i2 + 1 & i2 + 1 < sa..csize)" */
419         /*: assume "0 <= i2 & i2 < sa..csize" */
420         sa.remove_at(i2);
421
422         sb.remove_at(i2);
423         /*: note "(~(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < i2) & (i2, v1) :
                sa..(old contents) & 0 <= i2 & i2 < sa..(old csize) & i2 < sa..(old csize) -
                1 & (i2 + 1, v1) : sa..(old contents) & 0 <= i2 + 1 & i2 + 1 < sa..(old
                csize)) --> (i2, v1) : sb..contents" */
424         int r1b = sb.indexOf(v1);
425
426         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
427     }
428
429     static void indexOf_remove_at_between_c_70(ArrayList sa, ArrayList sb, Object v1,
            int i2)
430     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
431                     sa..contents = sb..contents & sa..csize = sb..csize"
432         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
433         ensures "True" */
434     {
435         int r1a = sa.indexOf(v1);
436         /*: assume "~(r1a < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize - 1
                & (i2 + 1, v1) : sa..contents & 0 <= i2 + 1 & i2 + 1 < sa..csize))" */
437         /*: assume "0 <= i2 & i2 < sa..csize" */
438         sa.remove_at(i2);
439
440         /*: assume "0 <= i2 & i2 < sb..csize" */
441         sb.remove_at(i2);
442         /*: ghost specvar sb__contents :: "(int * obj) set" = "sb..contents" */
443         int r1b = sb.indexOf(v1);
444
445         {
446             /*: assuming "(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < sa..(old
                    csize)) & ~(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < i2) & (i2,
                    v1) ~: sa..(old contents)" */
447             /*: note "(r1a - 1, v1) : sb__contents" */
448         }
449
450         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
451     }
452
453     static void indexOf_remove_at_post_s_71(ArrayList sa, ArrayList sb, Object v1, int
            i2)
454     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
```

```
455                 sa..contents = sb..contents & sa..csize = sb..csize"
456         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
457         ensures "True" */
458     {
459         int r1a = sa.indexOf(v1);
460         /*: assume "0 <= i2 & i2 < sa..csize" */
461         sa.remove_at(i2);
462         /*: assume "r1a < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize & (i2,
                v1) : sa..contents & 0 <= i2 & i2 < sa..csize)" */
463 
464         sb.remove_at(i2);
465         /*: note "(~(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < i2) & (i2, v1) :
                sa..(old contents) & 0 <= i2 & i2 < sa..(old csize) & i2 < sa..(old csize) -
                1 & (i2 + 1, v1) : sa..(old contents) & 0 <= i2 + 1 & i2 + 1 < sa..(old
                csize)) --> (i2, v1) : sb..contents" */
466         int r1b = sb.indexOf(v1);
467 
468         /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
469     }
470 
471     static void indexOf_remove_at_post_c_71(ArrayList sa, ArrayList sb, Object v1, int
            i2)
472     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
473                 sa..contents = sb..contents & sa..csize = sb..csize"
474         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
475         ensures "True" */
476     {
477         int r1a = sa.indexOf(v1);
478         /*: assume "0 <= i2 & i2 < sa..csize" */
479         sa.remove_at(i2);
480         /*: assume "~(r1a < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize &
                (i2, v1) : sa..contents & 0 <= i2 & i2 < sa..csize))" */
481 
482         /*: assume "0 <= i2 & i2 < sb..csize" */
483         sb.remove_at(i2);
484         /*: ghost specvar sb__contents :: "(int * obj) set" = "sb..contents" */
485         int r1b = sb.indexOf(v1);
486 
487         {
488             /*: assuming "(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < sa..(old
                    csize)) & ~(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < i2) & (i2,
                    v1) ~: sa..(old contents)" */
489             /*: note "(r1a - 1, v1) : sb__contents" */
490         }
491 
492         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
493     }
494 
495     static void lastIndexOf_add_at_pre_c_81(ArrayList sa, ArrayList sb, Object v1, int
            i2, Object v2)
496     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
497                 sa..contents = sb..contents & sa..csize = sb..csize"
498         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
499         ensures "True" */
500     {
501         /*: assume "~((~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
                v2) | ((EX i. (i, v1) : sa..contents & 0 <= i & i < i2) & ~(EX i. (i, v1) :
                sa..contents & i2 <= i & i < sa..csize) & v1 ~= v2))" */
502         int r1a = sa.lastIndexOf(v1);
503         /*: assume "0 <= i2 & i2 <= sa..csize" */
504         sa.add_at(i2, v2);
505 
506         /*: assume "0 <= i2 & i2 <= sb..csize" */
```

```
507         sb.add_at(i2, v2);
508         /*: ghost specvar sb__contents :: "(int * obj) set" = "sb..contents" */
509         int r1b = sb.lastIndexOf(v1);
510 
511         {
512             /*: assuming "(EX i. (i, v1) : sa..(old contents) & i2 <= i & i < sa..(old
                    csize))" */
513             /*: note "(r1a + 1, v1) : sb__contents" */
514         }
515 
516         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
517     }
518 
519     static void lastIndexOf_add_at_between_c_82(ArrayList sa, ArrayList sb, Object v1,
            int i2, Object v2)
520     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
521                   sa..contents = sb..contents & sa..csize = sb..csize"
522         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
523         ensures "True" */
524     {
525         int r1a = sa.lastIndexOf(v1);
526         /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2))" */
527         /*: assume "0 <= i2 & i2 <= sa..csize" */
528         sa.add_at(i2, v2);
529 
530         /*: assume "0 <= i2 & i2 <= sb..csize" */
531         sb.add_at(i2, v2);
532         /*: ghost specvar sb__contents :: "(int * obj) set" = "sb..contents" */
533         int r1b = sb.lastIndexOf(v1);
534 
535         {
536             /*: assuming "(EX i. (i, v1) : sa..(old contents) & i2 <= i & i < sa..(old
                    csize))" */
537             /*: note "(r1a + 1, v1) : sb__contents" */
538         }
539 
540         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
541     }
542 
543     static void lastIndexOf_add_at_post_c_83(ArrayList sa, ArrayList sb, Object v1, int
            i2, Object v2)
544     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
545                   sa..contents = sb..contents & sa..csize = sb..csize"
546         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
547         ensures "True" */
548     {
549         int r1a = sa.lastIndexOf(v1);
550         /*: assume "0 <= i2 & i2 <= sa..csize" */
551         sa.add_at(i2, v2);
552         /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2))" */
553 
554         /*: assume "0 <= i2 & i2 <= sb..csize" */
555         sb.add_at(i2, v2);
556         /*: ghost specvar sb__contents :: "(int * obj) set" = "sb..contents" */
557         int r1b = sb.lastIndexOf(v1);
558 
559         {
560             /*: assuming "(EX i. (i, v1) : sa..(old contents) & i2 <= i & i < sa..(old
                    csize))" */
561             /*: note "(r1a + 1, v1) : sb__contents" */
562         }
```

```
563
564         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
565     }
566
567     static void remove_at_indexOf_pre_s_114(ArrayList sa, ArrayList sb, int i1, Object
            v2)
568     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
569                     sa..contents = sb..contents & sa..csize = sb..csize"
570         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
571         ensures "True" */
572     {
573         /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
                (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) : sa..contents
                & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & i1
                < sa..csize - 1 & (i1 + 1, v2) : sa..contents & 0 <= i1 + 1 & i1 + 1 <
                sa..csize)" */
574         /*: assume "0 <= i1 & i1 < sa..csize" */
575         Object r1a = sa.remove_at(i1);
576         /*: note "(~(EX i. (i, v2) : sa..(old contents) & 0 <= i & i < i1) & (i1, v2) :
                sa..(old contents) & 0 <= i1 & i1 < sa..(old csize) & i1 < sa..(old csize) -
                1 & (i1 + 1, v2) : sa..(old contents) & 0 <= i1 + 1 & i1 + 1 < sa..(old
                csize)) --> (i1, v2) : sa..contents" */
577         int r2a = sa.indexOf(v2);
578
579         int r2b = sb.indexOf(v2);
580         Object r1b = sb.remove_at(i1);
581
582         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
583     }
584
585     static void remove_at_indexOf_pre_c_114(ArrayList sa, ArrayList sb, int i1, Object
            v2)
586     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
587                     sa..contents = sb..contents & sa..csize = sb..csize"
588         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
589         ensures "True" */
590     {
591         /*: assume "~(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
                (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) : sa..contents
                & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & i1
                < sa..csize - 1 & (i1 + 1, v2) : sa..contents & 0 <= i1 + 1 & i1 + 1 <
                sa..csize))" */
592         /*: assume "0 <= i1 & i1 < sa..csize" */
593         Object r1a = sa.remove_at(i1);
594         /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
595         int r2a = sa.indexOf(v2);
596
597         int r2b = sb.indexOf(v2);
598         /*: assume "0 <= i1 & i1 < sb..csize" */
599         Object r1b = sb.remove_at(i1);
600
601         {
602             /*: assuming "(EX i. (i, v2) : sa..(old contents) & 0 <= i & i < sa..(old
                    csize)) & ~(EX i. (i, v2) : sa..(old contents) & 0 <= i & i < i1) & (i1,
                    v2) ~: sa..(old contents)" */
603             /*: note "(r2b - 1, v2) : sa__contents" */
604         }
605
606         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
607     }
608
```

```
609     static void remove_at_indexOf_between_s_115(ArrayList sa, ArrayList sb, int i1,
            Object v2)
610     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
611                 sa..contents = sb..contents & sa..csize = sb..csize"
612         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
613         ensures "True" */
614     {
615         /*: assume "0 <= i1 & i1 < sa..csize" */
616         Object r1a = sa.remove_at(i1);
617         /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & r1a ~=
                v2) | (EX i. (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) :
                sa..contents & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 <
                sa..csize & r1a = v2 & i1 < sa..csize)" */
618         {
619             /*: assuming "(EX i. (i, v2) : sa..(old contents) & i1 + 1 <= i & i <
                    sa..(old csize))" */
620             {
621                 /*: pickWitness j :: int suchThat "(j, v2) : sa..(old contents) & i1 +
                        1 <= j & j < sa..(old csize)" */
622                 /*: note "(j - 1, v2) : sa..contents & i1 <= j - 1 & j - 1 < sa..csize"
                        */
623                 /*: note "(EX i. (i, v2) : sa..contents & i1 <= i & i < sa..csize)" */
624             }
625             /*: note "(EX i. (i, v2) : sa..contents & i1 <= i & i < sa..csize)" */
626         }
627         int r2a = sa.indexOf(v2);
628
629         int r2b = sb.indexOf(v2);
630         Object r1b = sb.remove_at(i1);
631
632         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
633     }
634
635     static void remove_at_indexOf_between_c_115(ArrayList sa, ArrayList sb, int i1,
            Object v2)
636     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
637                 sa..contents = sb..contents & sa..csize = sb..csize"
638         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
639         ensures "True" */
640     {
641         /*: assume "0 <= i1 & i1 < sa..csize" */
642         Object r1a = sa.remove_at(i1);
643         /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
644         /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & r1a
                ~= v2) | (EX i. (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2)
                : sa..contents & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 <
                sa..csize & r1a = v2 & i1 < sa..csize))" */
645         int r2a = sa.indexOf(v2);
646
647         int r2b = sb.indexOf(v2);
648         /*: assume "0 <= i1 & i1 < sb..csize" */
649         Object r1b = sb.remove_at(i1);
650
651         {
652             /*: assuming "(EX i. (i, v2) : sa..(old contents) & 0 <= i & i < sa..(old
                    csize)) & ~(EX i. (i, v2) : sa..(old contents) & 0 <= i & i < i1) & (i1,
                    v2) ~: sa..(old contents)" */
653             /*: note "(r2b - 1, v2) : sa__contents" */
654         }
655
656         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize)" */
657     }
658
```

```
    static void remove_at_indexOf_post_s_116(ArrayList sa, ArrayList sb, int i1, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        {
            /*: assuming "(EX i. (i, v2) : sa..(old contents) & i1 + 1 <= i & i <
                sa..(old csize))" */
            {
                /*: pickWitness j :: int suchThat "(j, v2) : sa..(old contents) & i1 +
                    1 <= j & j < sa..(old csize)" */
                /*: note "(j - 1, v2) : sa..contents & i1 <= j - 1 & j - 1 < sa..csize"
                    */
                /*: note "(EX i. (i, v2) : sa..contents & i1 <= i & i < sa..csize)" */
            }
            /*: note "(EX i. (i, v2) : sa..contents & i1 <= i & i < sa..csize)" */
        }
        int r2a = sa.indexOf(v2);
        /*: assume "(r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1) | (r2a = i1 & r1a =
            v2 & i1 < sa..csize)" */

        int r2b = sb.indexOf(v2);
        Object r1b = sb.remove_at(i1);

        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize" */
    }

    static void remove_at_indexOf_post_c_116(ArrayList sa, ArrayList sb, int i1, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        Object r1a = sa.remove_at(i1);
        /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
        int r2a = sa.indexOf(v2);
        /*: assume "~((r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1) | (r2a = i1 & r1a =
            v2 & i1 < sa..csize))" */

        int r2b = sb.indexOf(v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        Object r1b = sb.remove_at(i1);

        {
            /*: assuming "(EX i. (i, v2) : sa..(old contents) & 0 <= i & i < sa..(old
                csize)) & ~(EX i. (i, v2) : sa..(old contents) & 0 <= i & i < i1) & (i1,
                v2) ~: sa..(old contents)" */
            /*: note "(r2b - 1, v2) : sa__contents" */
        }

        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
            sb..csize)" */
    }

    static void remove_at_lastIndexOf_between_s_118(ArrayList sa, ArrayList sb, int i1,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
```

```
712         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
713         ensures "True" */
714     {
715         /*: assume "0 <= i1 & i1 < sa..csize" */
716         Object r1a = sa.remove_at(i1);
717         /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & r1a ~=
                v2) | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
                sa..contents & i1 <= i & i < sa..csize) & r1a ~= v2)" */
718         {
719             /*: assuming "(EX i. (i, v2) : sa..(old contents) & i1 + 1 <= i & i <
                    sa..(old csize))" */
720             {
721                 /*: pickWitness j :: int suchThat "(j, v2) : sa..(old contents) & i1 +
                        1 <= j & j < sa..(old csize)" */
722                 /*: note "(j - 1, v2) : sa..contents & i1 <= j - 1 & j - 1 < sa..csize"
                        */
723                 /*: note "(EX i. (i, v2) : sa..contents & i1 <= i & i < sa..csize)" */
724             }
725             /*: note "(EX i. (i, v2) : sa..contents & i1 <= i & i < sa..csize)" */
726         }
727         int r2a = sa.lastIndexOf(v2);
728 
729         int r2b = sb.lastIndexOf(v2);
730         Object r1b = sb.remove_at(i1);
731 
732         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
733     }
734 
735     static void remove_at_lastIndexOf_post_s_119(ArrayList sa, ArrayList sb, int i1,
            Object v2)
736     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
737                   sa..contents = sb..contents & sa..csize = sb..csize"
738         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
739         ensures "True" */
740     {
741         /*: assume "0 <= i1 & i1 < sa..csize" */
742         Object r1a = sa.remove_at(i1);
743         {
744             /*: assuming "(EX i. (i, v2) : sa..(old contents) & i1 + 1 <= i & i <
                    sa..(old csize))" */
745             {
746                 /*: pickWitness j :: int suchThat "(j, v2) : sa..(old contents) & i1 +
                        1 <= j & j < sa..(old csize)" */
747                 /*: note "(j - 1, v2) : sa..contents & i1 <= j - 1 & j - 1 < sa..csize"
                        */
748                 /*: note "(EX i. (i, v2) : sa..contents & i1 <= i & i < sa..csize)" */
749             }
750             /*: note "(EX i. (i, v2) : sa..contents & i1 <= i & i < sa..csize)" */
751         }
752         int r2a = sa.lastIndexOf(v2);
753         /*: assume "(r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1 & r1a ~= v2)" */
754 
755         int r2b = sb.lastIndexOf(v2);
756         Object r1b = sb.remove_at(i1);
757 
758         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
                sb..csize" */
759     }
760 
761     static void remove_at_set_pre_c_129(ArrayList sa, ArrayList sb, int i1, int i2,
            Object v2)
762     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
763                   sa..contents = sb..contents & sa..csize = sb..csize"
764         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
```

```
765         ensures "True" */
766     {
767         /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
                sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) : sa..contents & 0
                <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & (ALL
                v. ((i1, v) : sa..contents) = ((i2 + 1, v) : sa..contents)) & (i1, v2) :
                sa..contents & (i2 + 1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <=
                i2 + 1 & i2 + 1 < sa..csize) | (i1 > i2))" */
768         /*: assume "0 <= i1 & i1 < sa..csize" */
769         Object r1a = sa.remove_at(i1);
770         /*: assume "0 <= i2 & i2 < sa..csize" */
771         sa.set(i2, v2);
772 
773         /*: assume "0 <= i2 & i2 < sb..csize" */
774         sb.set(i2, v2);
775         /*: assume "0 <= i1 & i1 < sb..csize" */
776         Object r1b = sb.remove_at(i1);
777 
778         {
779             /*: assuming "i1 < i2 & (i2, v2) ~: sa..(old contents)" */
780             /*: note "(i2 - 1, v2) ~: sa..contents" */
781             /*: note "(i2 - 1, v2) : sb..contents" */
782             /*: note "sa..contents ~= sb..contents" */
783         }
784 
785         /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
786     }
787 
788     static void remove_at_add_at_pre_c_135(ArrayList sa, ArrayList sb, int i1, int i2,
            Object v2)
789     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
790                   sa..contents = sb..contents & sa..csize = sb..csize"
791         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
792         ensures "True" */
793     {
794         /*: assume "~((i1 < i2 & (i2, v2) : sa..contents & 0 <= i2 & i2 < sa..csize) |
                (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize) | (i1 > i2 &
                (ALL v. ((i1 - 1, v) : sa..contents) = ((i1, v) : sa..contents)) & 0 <= i1 -
                1 & i1 - 1 < sa..csize & 0 <= i1 & i1 < sa..csize))" */
795         /*: assume "0 <= i1 & i1 < sa..csize" */
796         sa.remove_at(i1);
797         /*: assume "0 <= i2 & i2 <= sa..csize" */
798         sa.add_at(i2, v2);
799 
800         /*: assume "0 <= i2 & i2 <= sb..csize" */
801         sb.add_at(i2, v2);
802         /*: assume "0 <= i1 & i1 < sb..csize" */
803         sb.remove_at(i1);
804 
805         {
806             /*: assuming "i1 > i2 & ~(ALL v. ((i1 - 1, v) : sa..(old contents)) = ((i1,
                    v) : sa..(old contents)))" */
807             {
808                 /*: pickWitness w :: obj suchThat "((i1 - 1, w) ~: sa..(old contents))
                        & ((i1, w) : sa..(old contents))" */
809                 /*: note "(i1, w) ~: sa..contents" */
810                 /*: note "(i1, w) : sb..contents" */
811                 /*: note "sa..contents ~= sb..contents" */
812             }
813             /*: note "sa..contents ~= sb..contents" */
814         }
815 
816         /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
```

```
817     }
818
819     static void remove_at_add_at_between_c_136(ArrayList sa, ArrayList sb, int i1, int
            i2, Object v2)
820     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
821                 sa..contents = sb..contents & sa..csize = sb..csize"
822         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
                "sb..msize"
823         ensures "True" */
824     {
825         /*: assume "0 <= i1 & i1 < sa..csize" */
826         sa.remove_at(i1);
827         /*: assume "~((i1 < i2 & (i2 - 1, v2) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
                sa..csize) | (i1 = i2 & (i1, v2) : sa..(old contents) & 0 <= i1 & i1 <
                sa..(old csize)) | (i1 > i2 & (ALL v. ((i1 - 1, v) : sa..contents) = ((i1,
                v) : sa..(old contents))) & 0 <= i1 - 1 & i1 - 1 < sa..csize & 0 <= i1 & i1
                < sa..(old csize)))" */
828         /*: assume "0 <= i2 & i2 <= sa..csize" */
829         sa.add_at(i2, v2);
830
831         /*: assume "0 <= i2 & i2 <= sb..csize" */
832         sb.add_at(i2, v2);
833         /*: assume "0 <= i1 & i1 < sb..csize" */
834         sb.remove_at(i1);
835
836         {
837             /*: assuming "i1 > i2 & ~(ALL v. ((i1 - 1, v) : sa..(old contents)) = ((i1,
                    v) : sa..(old contents)))" */
838             {
839                 /*: pickWitness w :: obj suchThat "((i1 - 1, w) ~: sa..(old contents))
                        & ((i1, w) : sa..(old contents))" */
840                 /*: note "(i1, w) ~: sa..contents" */
841                 /*: note "(i1, w) : sb..contents" */
842                 /*: note "sa..contents ~= sb..contents" */
843             }
844             /*: note "sa..contents ~= sb..contents" */
845         }
846
847         /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
848     }
849
850     static void remove_at_indexOf_pre_s_141(ArrayList sa, ArrayList sb, int i1, Object
            v2)
851     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
852                 sa..contents = sb..contents & sa..csize = sb..csize"
853         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
854         ensures "True" */
855     {
856         /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
                (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) : sa..contents
                & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & i1
                < sa..csize - 1 & (i1 + 1, v2) : sa..contents & 0 <= i1 + 1 & i1 + 1 <
                sa..csize)" */
857         /*: assume "0 <= i1 & i1 < sa..csize" */
858         sa.remove_at(i1);
859         /*: note "(~(EX i. (i, v2) : sa..(old contents) & 0 <= i & i < i1) & (i1, v2) :
                sa..(old contents) & 0 <= i1 & i1 < sa..(old csize) & i1 < sa..(old csize) -
                1 & (i1 + 1, v2) : sa..(old contents) & 0 <= i1 + 1 & i1 + 1 < sa..(old
                csize)) --> (i1, v2) : sa..contents" */
860         int r2a = sa.indexOf(v2);
861
862         int r2b = sb.indexOf(v2);
863         sb.remove_at(i1);
864
865         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
```

```
    }

    static void remove_at_indexOf_pre_c_141(ArrayList sa, ArrayList sb, int i1, Object
        v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
            (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) : sa..contents
            & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & i1
            < sa..csize - 1 & (i1 + 1, v2) : sa..contents & 0 <= i1 + 1 & i1 + 1 <
            sa..csize))" */
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
        int r2a = sa.indexOf(v2);

        int r2b = sb.indexOf(v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.remove_at(i1);

        {
            /*: assuming "(EX i. (i, v2) : sa..(old contents) & 0 <= i & i < sa..(old
                csize)) & ~(EX i. (i, v2) : sa..(old contents) & 0 <= i & i < i1) & (i1,
                v2) ~: sa..(old contents)" */
            /*: note "(r2b - 1, v2) : sa__contents" */
        }

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void remove_at_indexOf_between_s_142(ArrayList sa, ArrayList sb, int i1,
        Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & (i1,
            v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | (EX i. (i, v2)
            : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) : sa..contents & 0 <= i
            & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & (i1, v2) :
            sa..(old contents) & 0 <= i1 & i1 < sa..(old csize) & i1 < sa..csize)" */
        {
            /*: assuming "(EX i. (i, v2) : sa..(old contents) & i1 + 1 <= i & i <
                sa..(old csize))" */
            {
                /*: pickWitness j :: int suchThat "(j, v2) : sa..(old contents) & i1 +
                    1 <= j & j < sa..(old csize)" */
                /*: note "(j - 1, v2) : sa..contents & i1 <= j - 1 & j - 1 < sa..csize"
                    */
                /*: note "(EX i. (i, v2) : sa..contents & i1 <= i & i < sa..csize)" */
            }
            /*: note "(EX i. (i, v2) : sa..contents & i1 <= i & i < sa..csize)" */
        }
        int r2a = sa.indexOf(v2);

        int r2b = sb.indexOf(v2);
        sb.remove_at(i1);

```

```
915         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
916     }
917 
918     static void remove_at_indexOf_between_c_142(ArrayList sa, ArrayList sb, int i1,
        Object v2)
919     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
920                   sa..contents = sb..contents & sa..csize = sb..csize"
921         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
922         ensures "True" */
923     {
924         /*: assume "0 <= i1 & i1 < sa..csize" */
925         sa.remove_at(i1);
926         /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
927         /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & (i1,
            v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | (EX i. (i, v2)
            : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) : sa..contents & 0 <= i
            & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & (i1, v2) :
            sa..(old contents) & 0 <= i1 & i1 < sa..(old csize) & i1 < sa..csize))" */
928         int r2a = sa.indexOf(v2);
929 
930         int r2b = sb.indexOf(v2);
931         /*: assume "0 <= i1 & i1 < sb..csize" */
932         sb.remove_at(i1);
933 
934         {
935             /*: assuming "(EX i. (i, v2) : sa..(old contents) & 0 <= i & i < sa..(old
                csize)) & ~(EX i. (i, v2) : sa..(old contents) & 0 <= i & i < i1) & (i1,
                v2) ~: sa..(old contents)" */
936             /*: note "(r2b - 1, v2) : sa__contents" */
937         }
938 
939         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
940     }
941 
942     static void remove_at_indexOf_post_s_143(ArrayList sa, ArrayList sb, int i1, Object
        v2)
943     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
944                   sa..contents = sb..contents & sa..csize = sb..csize"
945         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
946         ensures "True" */
947     {
948         /*: assume "0 <= i1 & i1 < sa..csize" */
949         sa.remove_at(i1);
950         {
951             /*: assuming "(EX i. (i, v2) : sa..(old contents) & i1 + 1 <= i & i <
                sa..(old csize))" */
952             {
953                 /*: pickWitness j :: int suchThat "(j, v2) : sa..(old contents) & i1 +
                    1 <= j & j < sa..(old csize)" */
954                 /*: note "(j - 1, v2) : sa..contents & i1 <= j - 1 & j - 1 < sa..csize"
                    */
955                 /*: note "(EX i. (i, v2) : sa..contents & i1 <= i & i < sa..csize)" */
956             }
957             /*: note "(EX i. (i, v2) : sa..contents & i1 <= i & i < sa..csize)" */
958         }
959         int r2a = sa.indexOf(v2);
960         /*: assume "(r2a < 0 & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old
            csize)) | (0 <= r2a & r2a < i1) | (r2a = i1 & (i1, v2) : sa..(old contents)
            & 0 <= i1 & i1 < sa..(old csize) & i1 < sa..csize)" */
961 
962         int r2b = sb.indexOf(v2);
963         sb.remove_at(i1);
964 
965         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
```

```
966     }
967
968     static void remove_at_indexOf_post_c_143(ArrayList sa, ArrayList sb, int i1, Object
            v2)
969     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
970                 sa..contents = sb..contents & sa..csize = sb..csize"
971         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
972         ensures "True" */
973     {
974         /*: assume "0 <= i1 & i1 < sa..csize" */
975         sa.remove_at(i1);
976         /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
977         int r2a = sa.indexOf(v2);
978         /*: assume "~((r2a < 0 & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 <
                sa..(old csize)) | (0 <= r2a & r2a < i1) | (r2a = i1 & (i1, v2) : sa..(old
                contents) & 0 <= i1 & i1 < sa..(old csize) & i1 < sa..csize))" */
979
980         int r2b = sb.indexOf(v2);
981         /*: assume "0 <= i1 & i1 < sb..csize" */
982         sb.remove_at(i1);
983
984         {
985             /*: assuming "(EX i. (i, v2) : sa..(old contents) & 0 <= i & i < sa..(old
                    csize)) & ~(EX i. (i, v2) : sa..(old contents) & 0 <= i & i < i1) & (i1,
                    v2) ~: sa..(old contents)" */
986             /*: note "(r2b - 1, v2) : sa__contents" */
987         }
988
989         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
                */
990     }
991
992     static void remove_at_lastIndexOf_between_s_145(ArrayList sa, ArrayList sb, int i1,
            Object v2)
993     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
994                 sa..contents = sb..contents & sa..csize = sb..csize"
995         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
996         ensures "True" */
997     {
998         /*: assume "0 <= i1 & i1 < sa..csize" */
999         sa.remove_at(i1);
1000        /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & (i1,
                v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | ((EX i. (i,
                v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) : sa..contents & i1
                <= i & i < sa..csize) & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 <
                sa..(old csize))" */
1001        {
1002            /*: assuming "(EX i. (i, v2) : sa..(old contents) & i1 + 1 <= i & i <
                    sa..(old csize))" */
1003            {
1004                /*: pickWitness j :: int suchThat "(j, v2) : sa..(old contents) & i1 +
                        1 <= j & j < sa..(old csize)" */
1005                /*: note "(j - 1, v2) : sa..contents & i1 <= j - 1 & j - 1 < sa..csize"
                        */
1006                /*: note "(EX i. (i, v2) : sa..contents & i1 <= i & i < sa..csize)" */
1007            }
1008            /*: note "(EX i. (i, v2) : sa..contents & i1 <= i & i < sa..csize)" */
1009        }
1010        int r2a = sa.lastIndexOf(v2);
1011
1012        int r2b = sb.lastIndexOf(v2);
1013        sb.remove_at(i1);
1014
1015        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
1016    }
```

```
1017
1018    static void remove_at_lastIndexOf_post_s_146(ArrayList sa, ArrayList sb, int i1,
            Object v2)
1019    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1020                sa..contents = sb..contents & sa..csize = sb..csize"
1021        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1022        ensures "True" */
1023    {
1024        /*: assume "0 <= i1 & i1 < sa..csize" */
1025        sa.remove_at(i1);
1026        {
1027            /*: assuming "(EX i. (i, v2) : sa..(old contents) & i1 + 1 <= i & i <
                    sa..(old csize))" */
1028            {
1029                /*: pickWitness j :: int suchThat "(j, v2) : sa..(old contents) & i1 +
                        1 <= j & j < sa..(old csize)" */
1030                /*: note "(j - 1, v2) : sa..contents & i1 <= j - 1 & j - 1 < sa..csize"
                        */
1031                /*: note "(EX i. (i, v2) : sa..contents & i1 <= i & i < sa..csize)" */
1032            }
1033            /*: note "(EX i. (i, v2) : sa..contents & i1 <= i & i < sa..csize)" */
1034        }
1035        int r2a = sa.lastIndexOf(v2);
1036        /*: assume "(r2a < 0 & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old
                csize)) | (0 <= r2a & r2a < i1 & (i1, v2) ~: sa..(old contents) & 0 <= i1 &
                i1 < sa..(old csize))" */
1037
1038        int r2b = sb.lastIndexOf(v2);
1039        sb.remove_at(i1);
1040
1041        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
1042    }
1043
1044    static void remove_at_remove_at_pre_c_147(ArrayList sa, ArrayList sb, int i1, int
            i2)
1045    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1046                sa..contents = sb..contents & sa..csize = sb..csize"
1047        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1048        ensures "True" */
1049    {
1050        /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
                sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
                sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :
                sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
                sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
                sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
                <= i1 + 1 & i1 + 1 < sa..csize))" */
1051        /*: assume "0 <= i1 & i1 < sa..csize" */
1052        sa.remove_at(i1);
1053        /*: assume "0 <= i2 & i2 < sa..csize" */
1054        Object r2a = sa.remove_at(i2);
1055
1056        /*: assume "0 <= i2 & i2 < sb..csize" */
1057        Object r2b = sb.remove_at(i2);
1058        /*: assume "0 <= i1 & i1 < sb..csize" */
1059        sb.remove_at(i1);
1060
1061        {
1062            /*: assuming "i1 > i2 & ~(ALL v. ((i1, v) : sa..(old contents)) = ((i1 + 1,
                    v) : sa..(old contents)))" */
1063            {
1064                /*: pickWitness w :: obj suchThat "((i1, w) : sa..(old contents)) &
                        ((i1 + 1, w) ~: sa..(old contents))" */
1065                /*: note "(i1 - 1, w) ~: sa..contents" */
1066                /*: note "(i1 - 1, w) : sb..contents" */
```

```
            /*: note "sa..contents ~= sb..contents" */
        }
        /*: note "sa..contents ~= sb..contents" */
    }

    /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
}

static void remove_at_remove_at_between_c_148(ArrayList sa, ArrayList sb, int i1,
    int i2)
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
              sa..contents = sb..contents & sa..csize = sb..csize"
    modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
    ensures "True" */
{
    /*: assume "0 <= i1 & i1 < sa..csize" */
    sa.remove_at(i1);
    /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
        sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
        sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..(old contents)) = ((i2, v) :
        sa..contents)) & 0 <= i1 & i1 < sa..(old csize) & 0 <= i2 & i2 < sa..csize)
        | (sa..csize > i1 & i1 > i2 & (ALL v. ((i1, v) : sa..(old contents)) = ((i1,
        v) : sa..contents)) & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 & i1 <
        sa..csize))" */
    /*: assume "0 <= i2 & i2 < sa..csize" */
    Object r2a = sa.remove_at(i2);

    /*: assume "0 <= i2 & i2 < sb..csize" */
    Object r2b = sb.remove_at(i2);
    /*: assume "0 <= i1 & i1 < sb..csize" */
    sb.remove_at(i1);

    {
        /*: assuming "i1 > i2 & ~(ALL v. ((i1, v) : sa..(old contents)) = ((i1 + 1,
            v) : sa..(old contents)))" */
        {
            /*: pickWitness w :: obj suchThat "((i1, w) : sa..(old contents)) &
                ((i1 + 1, w) ~: sa..(old contents))" */
            /*: note "(i1 - 1, w) ~: sa..contents" */
            /*: note "(i1 - 1, w) : sb..contents" */
            /*: note "sa..contents ~= sb..contents" */
        }
        /*: note "sa..contents ~= sb..contents" */
    }

    /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
}

static void remove_at_remove_at_pre_c_150(ArrayList sa, ArrayList sb, int i1, int
    i2)
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
              sa..contents = sb..contents & sa..csize = sb..csize"
    modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
    ensures "True" */
{
    /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
        sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
        sa..csize) | i1 = i2 | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
        sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
        <= i1 + 1 & i1 + 1 < sa..csize))" */
    /*: assume "0 <= i1 & i1 < sa..csize" */
    sa.remove_at(i1);
    /*: assume "0 <= i2 & i2 < sa..csize" */
```

```
        sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.remove_at(i1);

        {
            /*: assuming "i1 < i2 & ~(ALL v. ((i2, v) : sa..(old contents)) = ((i2 + 1,
                v) : sa..(old contents)))" */
            {
                /*: pickWitness w :: obj suchThat "((i2, w) : sa..(old contents)) &
                    ((i2 + 1, w) ~: sa..(old contents))" */
                /*: note "(i2 - 1, w) : sa..contents" */
                /*: note "(i2 - 1, w) ~: sb..contents" */
                /*: note "sa..contents ~= sb..contents" */
            }
            /*: note "sa..contents ~= sb..contents" */
        }
        {
            /*: assuming "i1 > i2 & ~(ALL v. ((i1, v) : sa..(old contents)) = ((i1 + 1,
                v) : sa..(old contents)))" */
            {
                /*: pickWitness w :: obj suchThat "((i1, w) : sa..(old contents)) &
                    ((i1 + 1, w) ~: sa..(old contents))" */
                /*: note "(i1 - 1, w) ~: sa..contents" */
                /*: note "(i1 - 1, w) : sb..contents" */
                /*: note "sa..contents ~= sb..contents" */
            }
            /*: note "sa..contents ~= sb..contents" */
        }

        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
    }

    static void remove_at_remove_at_between_c_151(ArrayList sa, ArrayList sb, int i1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.remove_at(i1);
        /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
            sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
            sa..csize) | i1 = i2 | (sa..csize > i1 & i1 > i2 & (ALL v. ((i1, v) :
            sa..(old contents)) = ((i1, v) : sa..contents)) & 0 <= i1 & i1 < sa..(old
            csize) & 0 <= i1 & i1 < sa..csize))" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.remove_at(i1);

        {
            /*: assuming "i1 < i2 & ~(ALL v. ((i2, v) : sa..(old contents)) = ((i2 + 1,
                v) : sa..(old contents)))" */
            {
                /*: pickWitness w :: obj suchThat "((i2, w) : sa..(old contents)) &
                    ((i2 + 1, w) ~: sa..(old contents))" */
                /*: note "(i2 - 1, w) : sa..contents" */
                /*: note "(i2 - 1, w) ~: sb..contents" */
```

```
            /*: note "sa..contents ~= sb..contents" */
        }
        /*: note "sa..contents ~= sb..contents" */
    }
    {
        /*: assuming "i1 > i2 & ~(ALL v. ((i1, v) : sa..(old contents)) = ((i1 + 1,
            v) : sa..(old contents)))" */
        {
            /*: pickWitness w :: obj suchThat "((i1, w) : sa..(old contents)) &
                ((i1 + 1, w) ~: sa..(old contents))" */
            /*: note "(i1 - 1, w) ~: sa..contents" */
            /*: note "(i1 - 1, w) : sb..contents" */
            /*: note "sa..contents ~= sb..contents" */
        }
        /*: note "sa..contents ~= sb..contents" */
    }

    /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
}

static void remove_at_set_pre_c_156(ArrayList sa, ArrayList sb, int i1, int i2,
    Object v2)
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
              sa..contents = sb..contents & sa..csize = sb..csize"
    modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
    ensures "True" */
{
    /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
        sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) : sa..contents & 0
        <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & (i2
        + 1, v2) : sa..contents & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 > i2))" */
    /*: assume "0 <= i1 & i1 < sa..csize" */
    sa.remove_at(i1);
    /*: assume "0 <= i2 & i2 < sa..csize" */
    sa.set(i2, v2);

    /*: assume "0 <= i2 & i2 < sb..csize" */
    sb.set(i2, v2);
    /*: assume "0 <= i1 & i1 < sb..csize" */
    sb.remove_at(i1);

    {
        /*: assuming "i1 < i2 & (i2, v2) ~: sa..(old contents)" */
        /*: note "(i2 - 1, v2) ~: sa..contents" */
        /*: note "(i2 - 1, v2) : sb..contents" */
        /*: note "sa..contents ~= sb..contents" */
    }

    /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
}

static void set_add_at_pre_c_189(ArrayList sa, ArrayList sb, int i1, Object v1, int
    i2, Object v2)
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
              sa..contents = sb..contents & sa..csize = sb..csize"
    modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
        "sb..msize"
    ensures "True" */
{
    /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
        sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | (i1 > i2 & (ALL v. ((i1
        - 1, v) : sa..contents) = ((i1, v) : sa..contents)) & (i1 - 1, v1) :
        sa..contents & (i1, v1) : sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize &
        0 <= i1 & i1 < sa..csize))" */
    /*: assume "0 <= i1 & i1 < sa..csize" */
```

```
        sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        /*: assume "0 <= i2 & i2 <= sb..csize" */
        sb.add_at(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.set(i1, v1);

        {
            /*: assuming "i1 >= i2 & (i1, v1) ~: sa..(old contents)" */
            /*: note "(i1 + 1, v1) : sa..contents" */
            /*: note "sa..contents ~= sb..contents" */
        }

        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
    }

    static void set_add_at_between_c_190(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.set(i1, v1);
        /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & (i1, v2) :
            sa..(old contents) & v1 = v2 & 0 <= i1 & i1 < sa..(old csize)) | (i1 > i2 &
            (ALL v. ((i1 - 1, v) : sa..contents) = ((i1, v) : sa..(old contents))) & (i1
            - 1, v1) : sa..contents & (i1, v1) : sa..(old contents) & 0 <= i1 - 1 & i1 -
            1 < sa..csize & 0 <= i1 & i1 < sa..(old csize)))" */
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);

        /*: assume "0 <= i2 & i2 <= sb..csize" */
        sb.add_at(i2, v2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.set(i1, v1);

        {
            /*: assuming "i1 >= i2 & (i1, v1) ~: sa..(old contents)" */
            /*: note "(i1 + 1, v1) : sa..contents" */
            /*: note "sa..contents ~= sb..contents" */
        }

        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
    }

    static void set_add_at_post_c_191(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2, Object v2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
            "sb..msize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 <= sa..csize" */
        sa.add_at(i2, v2);
        /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & (i1, v2) :
            sa..(old contents) & v1 = v2 & 0 <= i1 & i1 < sa..(old csize)) | (i1 > i2 &
            (ALL v. ((i1, v) : sa..contents) = ((i1, v) : sa..(old contents)) & (i1, v1)
```

```
        : sa..contents & (i1, v1) : sa..(old contents) & 0 <= i1 & i1 < sa..csize &
        0 <= i1 & i1 < sa..(old csize))))" */

    /*: assume "0 <= i2 & i2 <= sb..csize" */
    sb.add_at(i2, v2);
    /*: assume "0 <= i1 & i1 < sb..csize" */
    sb.set(i1, v1);

    {
        /*: assuming "i1 >= i2 & (i1, v1) ~: sa..(old contents)" */
        /*: note "(i1 + 1, v1) : sa..contents" */
        /*: note "sa..contents ~= sb..contents" */
    }

    /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
}

static void set_remove_at_pre_c_201(ArrayList sa, ArrayList sb, int i1, Object v1,
    int i2)
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
              sa..contents = sb..contents & sa..csize = sb..csize"
    modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
    ensures "True" */
{
    /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (ALL v. ((i1, v) :
        sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
        (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
        1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
        sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
        (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
        1 < sa..csize))" */
    /*: assume "0 <= i1 & i1 < sa..csize" */
    sa.set(i1, v1);
    /*: assume "0 <= i2 & i2 < sa..csize" */
    Object r2a = sa.remove_at(i2);

    /*: assume "0 <= i2 & i2 < sb..csize" */
    Object r2b = sb.remove_at(i2);
    /*: assume "0 <= i1 & i1 < sb..csize" */
    sb.set(i1, v1);

    {
        /*: assuming "i1 > i2 & (i1, v1) ~: sa..(old contents)" */
        /*: note "(i1 - 1, v1) : sa..contents" */
        /*: note "sa..contents ~= sb..contents" */
    }

    /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
}

static void set_remove_at_between_c_202(ArrayList sa, ArrayList sb, int i1, Object
    v1, int i2)
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
              sa..contents = sb..contents & sa..csize = sb..csize"
    modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
    ensures "True" */
{
    /*: assume "0 <= i1 & i1 < sa..csize" */
    sa.set(i1, v1);
    /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (ALL v. ((i1, v) :
        sa..(old contents)) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..(old
        contents) & (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) & 0
        <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v.
        ((i1, v) : sa..(old contents)) = ((i1 + 1, v) : sa..contents)) & (i1, v1) :
```

```
            sa..(old contents) & (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old
            csize) & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.set(i1, v1);

       {
            /*: assuming "i1 > i2 & (i1, v1) ~: sa..(old contents)" */
            /*: note "(i1 - 1, v1) : sa..contents" */
            /*: note "sa..contents ~= sb..contents" */
        }

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void set_remove_at_post_c_203(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "0 <= i1 & i1 < sa..csize" */
        sa.set(i1, v1);
        /*: assume "0 <= i2 & i2 < sa..csize" */
        Object r2a = sa.remove_at(i2);
        /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize & (ALL v. ((i1, v) : sa..(old
            contents)) = ((i1, v) : sa..contents)) & (i1, v1) : sa..(old contents) &
            (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 & i1 <
            sa..csize) | (sa..csize > i1 & i1 > i2 & (ALL v. ((i1, v) : sa..(old
            contents)) = ((i1, v) : sa..contents)) & (i1, v1) : sa..(old contents) &
            (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 & i1 <
            sa..csize))" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        Object r2b = sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.set(i1, v1);

        {
            /*: assuming "i1 > i2 & (i1, v1) ~: sa..(old contents)" */
            /*: note "(i1 - 1, v1) : sa..contents" */
            /*: note "sa..contents ~= sb..contents" */
        }

        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
            */
    }

    static void set_remove_at_pre_c_204(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2)
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
                  sa..contents = sb..contents & sa..csize = sb..csize"
        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
        ensures "True" */
    {
        /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, v1) :
            sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
            > i2 & (ALL v. ((i1, v) : sa..contents) = ((i1 + 1, v) : sa..contents)) &
```

```
            (i1, v1) : sa..contents & (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 <
            sa..csize & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
1378        /*: assume "0 <= i1 & i1 < sa..csize" */
1379        sa.set(i1, v1);
1380        /*: assume "0 <= i2 & i2 < sa..csize" */
1381        sa.remove_at(i2);
1382
1383        /*: assume "0 <= i2 & i2 < sb..csize" */
1384        sb.remove_at(i2);
1385        /*: assume "0 <= i1 & i1 < sb..csize" */
1386        sb.set(i1, v1);
1387
1388        {
1389            /*: assuming "i1 > i2 & (i1, v1) ~: sa..(old contents)" */
1390            /*: note "(i1 - 1, v1) : sa..contents" */
1391            /*: note "sa..contents ~= sb..contents" */
1392        }
1393
1394        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
1395    }
1396
1397    static void set_remove_at_between_c_205(ArrayList sa, ArrayList sb, int i1, Object
        v1, int i2)
1398    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1399                  sa..contents = sb..contents & sa..csize = sb..csize"
1400        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1401        ensures "True" */
1402    {
1403        /*: assume "0 <= i1 & i1 < sa..csize" */
1404        sa.set(i1, v1);
1405        /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, v1) :
            sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
            > i2 & (ALL v. ((i1, v) : sa..(old contents)) = ((i1 + 1, v) :
            sa..contents)) & (i1, v1) : sa..(old contents) & (i1 + 1, v1) : sa..contents
            & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
1406        /*: assume "0 <= i2 & i2 < sa..csize" */
1407        sa.remove_at(i2);
1408
1409        /*: assume "0 <= i2 & i2 < sb..csize" */
1410        sb.remove_at(i2);
1411        /*: assume "0 <= i1 & i1 < sb..csize" */
1412        sb.set(i1, v1);
1413
1414        {
1415            /*: assuming "i1 > i2 & (i1, v1) ~: sa..(old contents)" */
1416            /*: note "(i1 - 1, v1) : sa..contents" */
1417            /*: note "sa..contents ~= sb..contents" */
1418        }
1419
1420        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
1421    }
1422
1423    static void set_remove_at_post_c_206(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2)
1424    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1425                  sa..contents = sb..contents & sa..csize = sb..csize"
1426        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1427        ensures "True" */
1428    {
1429        /*: assume "0 <= i1 & i1 < sa..csize" */
1430        sa.set(i1, v1);
1431        /*: assume "0 <= i2 & i2 < sa..csize" */
1432        sa.remove_at(i2);
1433        /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize & (i1, v1) : sa..contents & 0
            <= i1 & i1 < sa..csize) | (sa..csize > i1 & i1 > i2 & (ALL v. ((i1, v) :
```

```
            sa..(old contents)) = ((i1, v) : sa..(contents))) & (i1, v1) : sa..(old
            contents) & (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) & 0 <=
            i1 & i1 < sa..csize))" */

        /*: assume "0 <= i2 & i2 < sb..csize" */
        sb.remove_at(i2);
        /*: assume "0 <= i1 & i1 < sb..csize" */
        sb.set(i1, v1);

        {
            /*: assuming "i1 > i2 & (i1, v1) ~: sa..(old contents)" */
            /*: note "(i1 - 1, v1) : sa..contents" */
            /*: note "sa..contents ~= sb..contents" */
        }

        /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
    }

}
```

B.6.3 Inverse Testing Methods

Listing 20. `ArrayListInv.java`

```
class ArrayListInv {
    static void add_at_0(ArrayList s, int i, Object v)
    /*: requires "s ~= null & s..init & 0 <= i & i <= s..csize"
        modifies "s..contents", "s..csize", "s..msize"
        ensures "True" */
    {
        s.add_at(i, v);
        s.remove_at(i);

        /*: assert "s..contents = s..(old contents) & s..csize = s..(old csize)" */
    }

    static void remove_at_1(ArrayList s, int i)
    /*: requires "s ~= null & s..init & 0 <= i & i < s..csize"
        modifies "s..contents", "s..csize", "s..msize"
        ensures "True" */
    {
        Object r = s.remove_at(i);
        s.add_at(i, r);

        /*: assert "s..contents = s..(old contents) & s..csize = s..(old csize)" */
    }

    static void set_2(ArrayList s, int i, Object v)
    /*: requires "s ~= null & s..init & 0 <= i & i < s..csize"
        modifies "s..contents"
        ensures "True" */
    {
        Object r = s.set(i, v);
        s.set(i, r);

        /*: assert "s..contents = s..(old contents) & s..csize = s..(old csize)" */
    }

}
```