



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

# 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
4     /*: public ghost specvar contents :: "obj set" = "{}"; */
5     /*: public specvar size :: "int"; */
6
7     private Node[] table;
8     private int _size;
9
10    public HashSet()
11    /*: modifies "init", "contents", "size"
12     ensures "init & contents = {} & size = 0" */ { ... }
13
14    public boolean add(Object v)
15    /*: requires "init & v ~= null"
16     modifies "contents", "size"
17     ensures "(v ~: old contents --> contents = old contents Un {v} & size = old size + 1 & result) &
18             (v : old contents --> contents = old contents & size = old size & ~result)" */ { ... }
19
20    public boolean contains(Object v)
21    /*: requires "init & v ~= null"
22     ensures "result = (v : contents)" */ { ... }
23
24    public boolean remove(Object v)
25    /*: requires "init & v ~= null"
26     modifies "contents", "size"
27     ensures "(v : old contents --> contents = old contents - {v} & size = old size - 1 & result) &
28             (v ~: old contents --> contents = old contents & size = old size & ~result)" */ { ... }
29
30    public int size()
31    /*: requires "init"
32     ensures "result = size" */ { ... }
33 }

```

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  $v1 = v2 \mid r1a$ ). 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

```

1  static void contains_add_between_s_40(HashSet sa, HashSet sb, Object v1, Object v2)
2  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null & v2 ~= null &
3     sa..contents = sb..contents & sa..size = sb..size"
4     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
5     ensures "True" */
6  {
7     boolean r1a = sa.contains(v1);
8     /*: assume "v1 ~= v2 | r1a" */
9     sa.add(v2);
10
11    sb.add(v2);
12    boolean r1b = sb.contains(v1);
13
14    /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
15 }
16
17 static void contains_add_between_c_40(HashSet sa, HashSet sb, Object v1, Object v2)
18 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null & v2 ~= null &
19    sa..contents = sb..contents & sa..size = sb..size"
20    modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
21    ensures "True" */
22 {
23    boolean r1a = sa.contains(v1);
24    /*: assume "~(v1 ~= v2 | r1a)" */
25    sa.add(v2);
26
27    sb.add(v2);
28    boolean r1b = sb.contains(v1);
29
30    /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
31 }

```

**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.

## 4. Formal Treatment

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

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.

```

1  static void method1_method2_(before | between | after)_c_id
2      (sa_decl, sb_decl, argv1_decls, argv2_decls)
3  /*: requires "sa ~= null & sb ~= null & sa ~= sb &
4     sa_abstract_state = sb_abstract_state"
5     modifies "sa_frame_condition", "sb_frame_condition"
6     ensures "True" */
7  {
8      /*: assume "~(before_commutativity_condition)" */
9      /*: assume "method1_precondition" */
10     r1a_type r1a = sa.method1(argv1);
11     /*: assume "~(between_commutativity_condition)" */
12     /*: assume "method2_precondition" */
13     r2a_type r2a = sa.method2(argv2);
14     /*: assume "~(after_commutativity_condition)" */
15
16     /*: assume "method2_precondition" */
17     r2b_type r2b = sb.method2(argv2);
18     /*: assume "method1_precondition" */
19     r1b_type r1b = sb.method1(argv1);
20
21     /*: assert "~(r1a = r1b & r2a = r2b &
22         sa_abstract_state = sb_abstract_state)" */
23 }

```

Figure 3. Template for Commutativity Testing Methods

```

1  static void method_id(s_decl, argv_decls)
2  /*: requires "s ~= null & method_precondition"
3     modifies "s_frame_condition"
4     ensures "True" */
5  {
6      r_type r = s.method(argv);
7      execute_inverse_operation();
8
9     /*: assert "s_abstract_state = s_initial_abstract_state" */
10 }

```

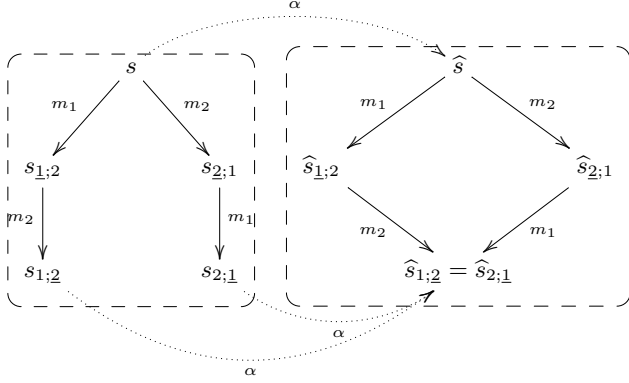
Figure 4. Template for Inverse Testing Methods

Given an operation  $m(v)$  on a given data structure,  $\text{pre}(m(v))$  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  $\alpha(s) \models \text{pre}(m(v))$  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):

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

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



**Figure 5.** Execution on Concrete States and Abstract States

can include the arguments  $v_1$  and  $v_2$ , the return values  $r_{1;2}$  and  $r_{2;1}$ , and abstract specification variables that denote various components of the three abstract states  $\alpha(s)$ ,  $\alpha(s_{1;2})$ , and  $\alpha(s_{2;1})$ . We write  $(\langle s_{1;2}, r_{1;2} \rangle = s.m_1(v_1); \langle s_{1;2}, r_{1;2} \rangle = s_{1;2}.m_2(v_2)) \models \phi$  if the commutativity condition  $\phi$  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  $\phi$  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  $\alpha(s) \models \text{pre}(m_1(v_1))$ ,  $\alpha(s_{1;2}) \models \text{pre}(m_2(v_2))$ , and  $(\langle s_{1;2}, r_{1;2} \rangle = s.m_1(v_1); \langle s_{1;2}, r_{1;2} \rangle = s_{1;2}.m_2(v_2)) \models \phi$  then  $\alpha(s) \models \text{pre}(m_2(v_2))$ ,  $\alpha(s_{2;1}) \models \text{pre}(m_1(v_1))$ ,  $r_{1;2} = r_{2;1}$ ,  $r_{1;2} = r_{2;1}$ , and  $\alpha(s_{1;2}) = \alpha(s_{2;1})$ .*

Given a commutativity condition  $\phi$  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  $\alpha(s) \models \text{pre}(m_1(v_1))$ ,  $\alpha(s_{1;2}) \models \text{pre}(m_2(v_2))$ , and  $(\langle s_{1;2}, r_{1;2} \rangle = s.m_1(v_1); \langle s_{1;2}, r_{1;2} \rangle = s_{1;2}.m_2(v_2)) \models \phi$  then  $\alpha(s) \models \text{pre}(m_2(v_2))$ ,  $\alpha(s_{2;1}) \models \text{pre}(m_1(v_1))$ ,  $r_{1;2} \neq r_{2;1}$ ,  $r_{1;2} \neq r_{2;1}$ , or  $\alpha(s_{1;2}) \neq \alpha(s_{2;1})$ .*

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  $\alpha(s_{1;2}) \models \text{pre}(m_2(v_2))$  and  $\alpha(s) = \alpha(s_{1;2})$ .*

## 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	
$s_1.\text{increase}(v_1)$	$s_2.\text{increase}(v_2)$	<i>true</i>	<i>true</i>
	$r_2 = s_2.\text{read}()$	$v_1 = 0$	$v_1 = 0$
$r_1 = s_1.\text{read}()$	$s_2.\text{increase}(v_2)$	$v_2 = 0$	$v_2 = 0$
	$r_2 = s_2.\text{read}()$	<i>true</i>	<i>true</i>

**Table 1.** Commutativity Conditions on Accumulator

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

**Table 2.** Before Commutativity Conditions on ListSet and HashSet

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

**Table 3.** Between Commutativity Conditions on ListSet and HashSet

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

**Table 4.** Before Commutativity Conditions on AssociationList and HashTable

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

**Table 5.** After Commutativity Conditions on AssociationList and HashTable



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

Table 6. Between Commutativity Conditions on ArrayList

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

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 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 <sup>1</sup>	12m 18s (3m 04s, 27s)

**Table 8.** Commutativity Testing Method Verification Times

Proof Language Command	Count
<code>note</code>	128
<code>assuming</code>	51
<code>pickWitness</code>	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  $(v1 = v2 \mid r1a)$  (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.

<sup>1</sup>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	$r = s.add(v)$	if $r = true$ then $s.remove(v)$
HashSet	$r = s.remove(v)$	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 enable 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. Proutzos, D. Nguyen, and K. Pingali. Defining and implementing commutativity conditions for parallel execution. Technical Report TR-ECE-09-11, School of Electrical 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	
$s_1.\text{increase}(v_1)$	$s_2.\text{increase}(v_2)$	<i>true</i>	<i>true</i>
	$r_2 = s_2.\text{read}()$	$v_1 = 0$	$v_1 = 0$
$r_1 = s_1.\text{read}()$	$s_2.\text{increase}(v_2)$	$v_2 = 0$	$v_2 = 0$
	$r_2 = s_2.\text{read}()$	<i>true</i>	<i>true</i>

### A.2 ListSet and HashSet

#### A.2.1 Before Commutativity Conditions

Table 12: Before Commutativity Conditions on ListSet and HashSet

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

#### A.2.2 Between Commutativity Conditions

Table 13: Between Commutativity Conditions on ListSet and HashSet

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



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

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

### A.3 AssociationList and HashTable

#### A.3.1 Before Commutativity Conditions

Table 15: Before Commutativity Conditions on AssociationList and HashTable

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

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

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

### A.3.2 Between Commutativity Conditions

Table 16: Between Commutativity Conditions on AssociationList and HashTable

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



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

	$r_2 = s_2.size()$	<i>true</i>	<i>true</i>
--	--------------------	-------------	-------------

## A.3.3 After Commutativity Conditions

Table 17: After Commutativity Conditions on AssociationList and HashTable

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

## A.4 ArrayList

### A.4.1 Before Commutativity Conditions

Table 18: Before Commutativity Conditions on ArrayList

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

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

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



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

	$r_2 = s_2.set(i_2, v_2)$	$( s_1  - 1 > i_1 > i_2 \wedge s_1[i_1] = s_1[i_1 + 1] = v_1)$ $i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1[i_1] = v_1 = v_2) \vee$ $i_1 > i_2$	$(s_1.size() - 1 > i_1 > i_2 \wedge s_1.get(i_1) = s_1.get(i_1 + 1) = v_1)$ $i_1 < i_2 \vee$ $(i_1 = i_2 \wedge s_1.get(i_1) = v_1 = v_2) \vee$ $i_1 > i_2$
	$s_2.set(i_2, v_2)$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge v_1 = v_2) \vee$ $i_1 > i_2$	$i_1 < i_2 \vee$ $(i_1 = i_2 \wedge v_1 = v_2) \vee$ $i_1 > i_2$
	$r_2 = s_2.size()$	<i>true</i>	<i>true</i>
$r_1 = s_1.size(v_1)$	$s_2.add.at(i_2, v_2)$	<i>false</i>	<i>false</i>
	$r_2 = s_2.get(i_2)$	<i>true</i>	<i>true</i>
	$r_2 = s_2.indexOf(v_2)$	<i>true</i>	<i>true</i>
	$r_2 = s_2.lastIndexOf(v_2)$	<i>true</i>	<i>true</i>
	$r_2 = s_2.remove.at(i_2)$	<i>false</i>	<i>false</i>
	$s_2.remove.at(i_2)$	<i>false</i>	<i>false</i>
	$r_2 = s_2.set(i_2, v_2)$	<i>true</i>	<i>true</i>
	$s_2.set(i_2, v_2)$	<i>true</i>	<i>true</i>
	$r_2 = s_2.size()$	<i>true</i>	<i>true</i>

## A.4.2 Between Commutativity Conditions

Table 19: Between Commutativity Conditions on ArrayList

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

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

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

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

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

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

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

## A.4.3 After Commutativity Conditions

Table 20: After Commutativity Conditions on ArrayList

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



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

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

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

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

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

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

## B. Source Code

### B.1 Accumulator

#### B.1.1 Data Structure

Listing 1. Accumulator.java

```
1 public class Accumulator {
2     private int accumulator;
3     /*: public specvar contents :: "int";
4         vardefs "contents == accumulator"; */
5
6     public Accumulator(int n)
7     /*: modifies "contents"
8         ensures "contents = n" */
9     {
10        accumulator = n;
11    }
12
13    public void increase(int n)
14    /*: modifies "contents"
15        ensures "contents = old contents + n" */
16    {
17        accumulator = accumulator + n;
18    }
19
20    public int read()
21    /*: ensures "result = contents" */
22    {
23        return accumulator;
24    }
25 }
```

#### B.1.2 Commutativity Testing Methods

Listing 2. AccumulatorComm.java

```
1 class AccumulatorComm {
2     static void increase_increase_pre_s_0(Accumulator sa, Accumulator sb, int n1, int
3         n2)
4     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
5         sa..contents = sb..contents"
6         modifies "sa..contents", "sb..contents"
7         ensures "True" */
8     {
9         /*: assume "True" */
10        sa.increase(n1);
11        sa.increase(n2);
12
13        sb.increase(n2);
14        sb.increase(n1);
15
16        /*: assert "sa..contents = sb..contents" */
17    }
18
19    static void increase_increase_pre_c_0(Accumulator sa, Accumulator sb, int n1, int
20        n2)
21    /*: requires "sa ~= null & sb ~= null & sa ~= sb &
22        sa..contents = sb..contents"
23        modifies "sa..contents", "sb..contents"
24        ensures "True" */
25    {
26        /*: assume "~(True)" */
27        sa.increase(n1);
28        sa.increase(n2);
29    }
```

```

28     sb.increase(n2);
29     sb.increase(n1);
30
31     /*: assert "~(sa..contents = sb..contents)" */
32 }
33
34 static void increase_increase_between_s_1(Accumulator sa, Accumulator sb, int n1,
35     int n2)
36 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
37     sa..contents = sb..contents"
38     modifies "sa..contents", "sb..contents"
39     ensures "True" */
40 {
41     sa.increase(n1);
42     /*: assume "True" */
43     sa.increase(n2);
44
45     sb.increase(n2);
46     sb.increase(n1);
47
48     /*: assert "sa..contents = sb..contents" */
49 }
50
51 static void increase_increase_between_c_1(Accumulator sa, Accumulator sb, int n1,
52     int n2)
53 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
54     sa..contents = sb..contents"
55     modifies "sa..contents", "sb..contents"
56     ensures "True" */
57 {
58     sa.increase(n1);
59     /*: assume "~(True)" */
60     sa.increase(n2);
61
62     sb.increase(n2);
63     sb.increase(n1);
64
65     /*: assert "~(sa..contents = sb..contents)" */
66 }
67
68 static void increase_increase_post_s_2(Accumulator sa, Accumulator sb, int n1, int
69     n2)
70 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
71     sa..contents = sb..contents"
72     modifies "sa..contents", "sb..contents"
73     ensures "True" */
74 {
75     sa.increase(n1);
76     sa.increase(n2);
77     /*: assume "True" */
78
79     sb.increase(n2);
80     sb.increase(n1);
81
82     /*: assert "sa..contents = sb..contents" */
83 }
84
85 static void increase_increase_post_c_2(Accumulator sa, Accumulator sb, int n1, int
86     n2)
87 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
88     sa..contents = sb..contents"
89     modifies "sa..contents", "sb..contents"
90     ensures "True" */
91 {
92     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     sb.increase(n2);
220     int r1b = sb.read();
221
222     /*: assert "~(ria = r1b & sa..contents = sb..contents)" */
223 }
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 "ria = 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 "~(ria = 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 "ria = 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 ~:
6           next..conts"
7        invariant ContsNull: "null..conts = {}" */
8 }

```

```

9 public class ListSet {
10     private Node head;
11     private int length;
12
13     /*: public specvar contents :: "obj set"
14         vardefs "contents == head..conts"
15
16         public specvar size :: "int"
17         vardefs "size == length"
18         invariant CardInv: "length = card (contents)"
19
20     private static specvar reachable :: "obj => obj => bool"
21     vardefs "reachable == (%x y. (x : Node & y = x..next) | (x : ListSet & y =
22         x..head))"
23     invariant InjInv: "ALL x1 x2 y. y ~= null & reachable x1 y & reachable x2 y -->
24         x1 = x2" */
25
26 public ListSet()
27 /*: modifies "contents", "size"
28     ensures "contents = {} & size = 0" */
29 {
30     head = null;
31     length = 0;
32
33     {
34         /*: pickAny x::obj */
35         {
36             /*: assuming CardHyp: "x : alloc & x : ListSet" */
37             {
38                 /*: assuming XIsThisHyp: "x = this" */
39                 /*: note SizeZero: "x..length = 0" */
40                 /*: note ContentsEmpty: "x..contents = {}" */
41                 /*: note XIsThisCard: "x..length = card (x..contents)" from
42                     XIsThisHyp, SizeZero, ContentsEmpty */
43             }
44             {
45                 /*: assuming XNotThisHyp: "x ~= this" */
46                 /*: note XInOldAlloc: "x : old alloc" */
47                 /*: note OldCard: "x..(old ListSet.length) = card (x..(old
48                     ListSet.contents))" from CardHyp, XNotThisHyp, XInOldAlloc,
49                     CardInv */
50                 /*: note XSizeEq: "x..length = x..(old ListSet.length)" */
51                 /*: note XContentsUnchanged: "x..contents = x..(old
52                     ListSet.contents)" */
53                 /*: note XNotThisCard: "x..length = card (x..contents)" from
54                     XSizeEq, OldCard, XContentsUnchanged */
55             }
56             /*: note CardConc: "x..length = card (x..contents)" from XIsThisCard,
57                 XNotThisCard */
58         }
59         /*: note CardPostCond: "x : alloc & x : ListSet --> x..length = card
60             (x..contents)" forSuch x */
61     }
62 }
63
64 public boolean add(Object v)
65 /*: requires "v ~= null"
66     modifies "contents", "size"
67     ensures "(v ~: old contents --> contents = old contents Un {v} & result & size
68         = old size + 1) & (v : old contents --> contents = old contents & ~result &
69         size = old size)" */
70 {
71     if (!_contains(v)) {
72         return false;
73     } else {

```

```

63     _add(v);
64     return true;
65 }
66 }
67
68 private void _add(Object v)
69 /*: requires "v ~= null & (v ~: contents) & theinvs"
70   modifies "new..Node.value", "new..Node.next", "new..Node.cnts", "head",
71     "length", "contents", "size"
72   ensures "contents = old contents Un {v} & size = old size + 1 & theinvs" */
73 {
74   Node n = new Node();
75   n.value = v;
76   n.next = head;
77   /*: "n..Node.cnts" := "{v} Un head..Node.cnts" */
78   head = n;
79
80   length = length + 1;
81
82   /*: note ContentsPost: "contents = old contents Un {v}" */
83   {
84     /*: pickAny x :: obj */
85     {
86       /*: assuming CardHyp: "x : alloc & x : ListSet" */
87       {
88         /*: note ThisProps: "this : old alloc & this : ListSet" */
89         /*: note OldCard: "old length = card (old contents)" from
90           ThisProps, CardInv */
91         /*: note NewSize: "length = old length + 1" */
92         /*: note NewNotInOld: "v ~: old contents" */
93         /*: note XIsThisCard: "length = card (contents)" from OldCard,
94           NewSize, NewNotInOld, ContentsPost */
95       }
96       {
97         /*: assuming XNotThisHyp: "x ~= this" */
98         /*: note XInOldAlloc: "x : old alloc" */
99         /*: note OldCard: "x..(old ListSet.length) = card (x..(old
100           ListSet.contents))" from CardHyp, XNotThisHyp, XInOldAlloc,
101           CardInv */
102         /*: note XSizeEq: "x..length = x..(old ListSet.length)" */
103         {
104           /*: localize */
105           /*: note XContentsForw: "ALL y. y : x..contents --> y : x..(old
106             ListSet.contents)" */
107           /*: note XContentsBack: "ALL y. y : x..(old ListSet.contents)
108             --> y : x..contents" */
109           /*: note XContentsUnchanged: "x..contents = x..(old
110             ListSet.contents)" from XContentsForw, XContentsBack */
111         }
112         /*: note XNotThisCard: "x..length = card (x..contents)" from
113           XSizeEq, OldCard, XContentsUnchanged */
114       }
115       /*: note CardConc: "x..length = card (x..contents)" from XIsThisCard,
116         XNotThisCard */
117     }
118     /*: note CardPostCond: "x : alloc & x : ListSet --> x..length = card
119       (x..contents)" forSuch x */
120   }
121 }
122
123 public boolean contains(Object v)
124 /*: ensures "result = (v : contents)" */
125 {
126   return _contains(v);
127 }

```

```

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 !=
124         null) {
125         if (curr.value == v) {
126             return true;
127         }
128         curr = curr.next;
129     }
130     return false;
131 }
132
133 public boolean remove(Object v)
134 /*: requires "v ~= null"
135 modifies "contents", "size"
136 ensures "(v : old contents --> contents = old contents - {v} & size = old size
137     - 1 & result) & (v ~: old contents --> contents = old contents & size = old
138     size & ~result)" */
139 {
140     if (_contains(v)) {
141         _remove(v);
142         return true;
143     } else {
144         return false;
145     }
146 }
147
148 private void _remove(Object v)
149 /*: requires "v ~= null & v : contents & theinvs"
150 modifies "Node.next", "Node.conts", "head", "length", "contents", "size"
151 ensures "contents = old contents - {v} & size = old size - 1 & theinvs" */
152 {
153     if (head.value == v) {
154         Node n = head.next;
155         head.next = null;
156         /*: "head..conts" := "{head..value}" */
157         head = n;
158     } else {
159         Node prev = head;
160         /*: "prev..conts" := "prev..conts - {v}" */
161         Node curr = prev.next;
162         while /*: invariant "prev..conts = prev..(old Node.conts) - {v} &
163             prev..next = curr & prev ~= curr & contents = old contents - {v} & (ALL
164             s. s : old Object.alloc & s : ListSet & s ~= this --> s..contents = old
165             (s..contents)) & v : curr..conts & (ALL n. n ~= null & n : Object.alloc
166             & n : Node & n ~= prev --> n..conts = {n..value} Un n..next..conts &
167             (n..value ~: n..next..conts)) & (ALL n. n..conts = n..(old conts) |
168             n..conts = n..(old conts) - {v})" */ (curr.value != v) {
169             /*: "curr..conts" := "curr..conts - {v}" */
170             prev = curr;
171             curr = curr.next;
172         }
173         Node n = curr.next;
174         prev.next = n;
175         curr.next = null;
176         /*: "curr..Node.conts" := "{curr..Node.value}" */
177     }
178
179     length = length - 1;
180
181     /*: note ContentsPost: "contents = old contents - {v}" */

```

```

173 {
174     /*: pickAny x::obj */
175     {
176         /*: assuming CardHyp: "x : alloc & x : ListSet" */
177         {
178             /*: note ThisProps: "this : old alloc & this : ListSet" */
179             /*: note OldCard: "old length = card (old contents)" from
180              ThisProps, CardInv */
181             /*: note NewSize: "length = old length - 1" */
182             /*: note NewNotInOld: "v : old contents" */
183             /*: note XIsThisCard: "length = card (contents)" from OldCard,
184              NewSize, NewNotInOld, ContentsPost */
185         }
186         {
187             /*: assuming XNotThisHyp: "x ~= this" */
188             /*: note XInOldAlloc: "x : old alloc" */
189             /*: note OldCard: "x..(old ListSet.length) = card (x..(old
190              ListSet.contents))" from CardHyp, XNotThisHyp, XInOldAlloc,
191              CardInv */
192             /*: note XSizeEq: "x..length = x..(old ListSet.length)" */
193             {
194                 /*: localize */
195                 /*: note XContentsForw: "ALL y. y : x..contents --> y : x..(old
196                  ListSet.contents)" */
197                 /*: note XContentsBack: "ALL y. y : x..(old ListSet.contents)
198                  --> y : x..contents" */
199                 /*: note XContentsUnchanged: "x..contents = x..(old
200                  ListSet.contents)" from XContentsForw, XContentsBack */
201             }
202             /*: note XNotThisCard: "x..length = card (x..contents)" from
203              XSizeEq, OldCard, XContentsUnchanged */
204         }
205         /*: note CardConc: "x..length = card (x..contents)" from XIsThisCard,
206          XNotThisCard */
207     }
208     /*: note CardPostCond: "x : alloc & x : ListSet --> x..length = card
209      (x..contents)" forSuch x */
210 }
211 }
212
213 public int size()
214 /*: ensures "result = size" */
215 {
216     return length;
217 }
218 }

```

## B.2.2 Commutativity Testing Methods

Listing 5. ListSetComm.java

```

1 class ListSetComm {
2     static void add_add_pre_s_0(ListSet sa, ListSet sb, Object v1, Object v2)
3     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
4      sa..contents = sb..contents & sa..size = sb..size"
5      modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
6      ensures "True" */
7     {
8         /*: assume "v1 ~= v2 | v1 : sa..contents" */
9         boolean r1a = sa.add(v1);
10        boolean r2a = sa.add(v2);
11
12        boolean r2b = sb.add(v2);
13        boolean r1b = sb.add(v1);
14    }

```

```

15     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
16         sb..size" */
17 }
18 static void add_add_pre_c_0(ListSet sa, ListSet sb, Object v1, Object v2)
19 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
20     sa..contents = sb..contents & sa..size = sb..size"
21     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
22     ensures "True" */
23 {
24     /*: assume "~(v1 ~= v2 | v1 : sa..contents)" */
25     boolean r1a = sa.add(v1);
26     boolean r2a = sa.add(v2);
27
28     boolean r2b = sb.add(v2);
29     boolean r1b = sb.add(v1);
30
31     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
32         sb..size)" */
33 }
34 static void add_add_between_s_1(ListSet sa, ListSet sb, Object v1, Object v2)
35 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
36     sa..contents = sb..contents & sa..size = sb..size"
37     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
38     ensures "True" */
39 {
40     boolean r1a = sa.add(v1);
41     /*: assume "v1 ~= v2 | ~r1a" */
42     boolean r2a = sa.add(v2);
43
44     boolean r2b = sb.add(v2);
45     boolean r1b = sb.add(v1);
46
47     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
48         sb..size" */
49 }
50 static void add_add_between_c_1(ListSet sa, ListSet sb, Object v1, Object v2)
51 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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 =
64         sb..size)" */
65 }
66 static void add_add_post_s_2(ListSet sa, ListSet sb, Object v1, Object v2)
67 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(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     sb.add(v2);
140     boolean r1b = sb.add(v1);
141
142     /*: assert "ria = r1b & sa..contents = sb..contents & sa..size = sb..size" */
143 }
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 "~(ria = 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 "ria = 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 "~(ria = 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"

```

```

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(ListSet sa, ListSet sb, Object v1, Object v2)
339 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(ListSet sa, ListSet sb, Object v1, Object v2)
355 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(ListSet sa, ListSet sb, Object v1, Object v2)
371 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(ListSet sa, ListSet sb, Object v1, Object v2)

```

```

387  /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(ListSet sa, ListSet sb, Object v1, Object v2)
403  /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(ListSet sa, ListSet sb, Object v1, Object v2)
419  /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(ListSet sa, ListSet sb, Object v1, Object v2)
435  /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(ListSet sa, ListSet sb, Object v1, Object v2)
451  /*: 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 =
496     sb..size" */
497 }
498
499 static void add_size_pre_c_15(ListSet sa, ListSet sb, Object v1)
500 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
501     sa..contents = sb..contents & sa..size = sb..size"
502     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
503     ensures "True" */
504 {
505     /*: assume "~(v1 : sa..contents)" */
506     boolean r1a = sa.add(v1);
507     int r2a = sa.size();
508
509     int r2b = sb.size();
510     boolean r1b = sb.add(v1);
511
512     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
513     sb..size)" */
514 }
515
516 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 &

```

```

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(ListSet sa, ListSet sb, Object v1, Object v2)
851 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= 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(ListSet sa, ListSet sb, Object v1, Object v2)
867 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(ListSet sa, ListSet sb, Object v1, Object v2)
883 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(ListSet sa, ListSet sb, Object v1, Object v2)
899 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(ListSet sa, ListSet sb, Object v1, Object v2)
915 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(ListSet sa, ListSet sb, Object v1, Object v2)
931 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(ListSet sa, ListSet sb, Object v1, Object v2)
947 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
948     sa..contents = sb..contents & sa..size = sb..size"
949     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
950     ensures "True" */
951 {
952     sa.add(v1);
953     boolean r2a = sa.remove(v2);
954     /*: assume "~(v1 ~= v2)" */
955
956     boolean r2b = sb.remove(v2);
957     sb.add(v1);
958
959     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
960 }
961
962 static void add_remove_pre_s_30(ListSet sa, ListSet sb, Object v1, Object v2)
963 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
964     sa..contents = sb..contents & sa..size = sb..size"
965     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"

```

```

966     ensures "True" */
967 {
968     /*: assume "v1 ~= v2" */
969     sa.add(v1);
970     sa.remove(v2);
971
972     sb.remove(v2);
973     sb.add(v1);
974
975     /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
976 }
977
978 static void add_remove_pre_c_30(ListSet sa, ListSet sb, Object v1, Object v2)
979 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
980     sa..contents = sb..contents & sa..size = sb..size"
981     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
982     ensures "True" */
983 {
984     /*: assume "~(v1 ~= v2)" */
985     sa.add(v1);
986     sa.remove(v2);
987
988     sb.remove(v2);
989     sb.add(v1);
990
991     /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
992 }
993
994 static void add_remove_between_s_31(ListSet sa, ListSet sb, Object v1, Object v2)
995 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
996     sa..contents = sb..contents & sa..size = sb..size"
997     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
998     ensures "True" */
999 {
1000     sa.add(v1);
1001     /*: assume "v1 ~= v2" */
1002     sa.remove(v2);
1003
1004     sb.remove(v2);
1005     sb.add(v1);
1006
1007     /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
1008 }
1009
1010 static void add_remove_between_c_31(ListSet sa, ListSet sb, Object v1, Object v2)
1011 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(ListSet sa, ListSet sb, Object v1, Object v2)
1027 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(ListSet sa, ListSet sb, Object v1, Object v2)
1043 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(ListSet sa, ListSet sb, Object v1)
1059 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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();
1069     sb.add(v1);
1070
1071     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
1072 }
1073
1074 static void add_size_pre_c_33(ListSet sa, ListSet sb, Object v1)
1075 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
1076     sa..contents = sb..contents & sa..size = sb..size"
1077     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1078     ensures "True" */
1079 {
1080     /*: assume "~(v1 : sa..contents)" */
1081     sa.add(v1);
1082     int r2a = sa.size();
1083
1084     int r2b = sb.size();
1085     sb.add(v1);
1086
1087     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
1088 }
1089
1090 static void add_size_between_s_34(ListSet sa, ListSet sb, Object v1)
1091 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
1092     sa..contents = sb..contents & sa..size = sb..size"
1093     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1094     ensures "True" */
1095 {

```

```

1096     sa.add(v1);
1097     /*: assume "v1 : sa..(old contents)" */
1098     int r2a = sa.size();
1099
1100     int r2b = sb.size();
1101     sb.add(v1);
1102
1103     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
1104 }
1105
1106 static void add_size_between_c_34(ListSet sa, ListSet sb, Object v1)
1107 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
1108             sa..contents = sb..contents & sa..size = sb..size"
1109    modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1110    ensures "True" */
1111 {
1112     sa.add(v1);
1113     /*: assume "~(v1 : sa..(old contents))" */
1114     int r2a = sa.size();
1115
1116     int r2b = sb.size();
1117     sb.add(v1);
1118
1119     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
1120 }
1121
1122 static void add_size_post_s_35(ListSet sa, ListSet sb, Object v1)
1123 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
1124             sa..contents = sb..contents & sa..size = sb..size"
1125    modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1126    ensures "True" */
1127 {
1128     sa.add(v1);
1129     int r2a = sa.size();
1130     /*: assume "v1 : sa..(old contents)" */
1131
1132     int r2b = sb.size();
1133     sb.add(v1);
1134
1135     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
1136 }
1137
1138 static void add_size_post_c_35(ListSet sa, ListSet sb, Object v1)
1139 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
1140             sa..contents = sb..contents & sa..size = sb..size"
1141    modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1142    ensures "True" */
1143 {
1144     sa.add(v1);
1145     int r2a = sa.size();
1146     /*: assume "~(v1 : sa..(old contents))" */
1147
1148     int r2b = sb.size();
1149     sb.add(v1);
1150
1151     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
1152 }
1153
1154 static void contains_add_pre_s_36(ListSet sa, ListSet sb, Object v1, Object v2)
1155 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
1156             sa..contents = sb..contents & sa..size = sb..size"
1157    modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1158    ensures "True" */
1159 {
1160     /*: assume "v1 ~= v2 | v1 : sa..contents" */

```



```

1161     boolean r1a = sa.contains(v1);
1162     boolean r2a = sa.add(v2);
1163
1164     boolean r2b = sb.add(v2);
1165     boolean r1b = sb.contains(v1);
1166
1167     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
        sb..size" */
1168 }
1169
1170 static void contains_add_pre_c_36(ListSet sa, ListSet sb, Object v1, Object v2)
1171 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
1172     sa..contents = sb..contents & sa..size = sb..size"
1173     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1174     ensures "True" */
1175 {
1176     /*: assume "~(v1 ~= v2 | v1 : sa..contents)" */
1177     boolean r1a = sa.contains(v1);
1178     boolean r2a = sa.add(v2);
1179
1180     boolean r2b = sb.add(v2);
1181     boolean r1b = sb.contains(v1);
1182
1183     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
        sb..size)" */
1184 }
1185
1186 static void contains_add_between_s_37(ListSet sa, ListSet sb, Object v1, Object v2)
1187 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
1188     sa..contents = sb..contents & sa..size = sb..size"
1189     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1190     ensures "True" */
1191 {
1192     boolean r1a = sa.contains(v1);
1193     /*: assume "v1 ~= v2 | r1a" */
1194     boolean r2a = sa.add(v2);
1195
1196     boolean r2b = sb.add(v2);
1197     boolean r1b = sb.contains(v1);
1198
1199     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
        sb..size" */
1200 }
1201
1202 static void contains_add_between_c_37(ListSet sa, ListSet sb, Object v1, Object v2)
1203 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
1204     sa..contents = sb..contents & sa..size = sb..size"
1205     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1206     ensures "True" */
1207 {
1208     boolean r1a = sa.contains(v1);
1209     /*: assume "~(v1 ~= v2 | r1a)" */
1210     boolean r2a = sa.add(v2);
1211
1212     boolean r2b = sb.add(v2);
1213     boolean r1b = sb.contains(v1);
1214
1215     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
        sb..size)" */
1216 }
1217
1218 static void contains_add_post_s_38(ListSet sa, ListSet sb, Object v1, Object v2)
1219 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
1220     sa..contents = sb..contents & sa..size = sb..size"
1221     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"

```

```

1222     ensures "True" */
1223 {
1224     boolean r1a = sa.contains(v1);
1225     boolean r2a = sa.add(v2);
1226     /*: assume "v1 ~= v2 | r1a" */
1227
1228     boolean r2b = sb.add(v2);
1229     boolean r1b = sb.contains(v1);
1230
1231     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
        sb..size" */
1232 }
1233
1234 static void contains_add_post_c_38(ListSet sa, ListSet sb, Object v1, Object v2)
1235 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
1236     sa..contents = sb..contents & sa..size = sb..size"
1237     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1238     ensures "True" */
1239 {
1240     boolean r1a = sa.contains(v1);
1241     boolean r2a = sa.add(v2);
1242     /*: assume "~(v1 ~= v2 | r1a)" */
1243
1244     boolean r2b = sb.add(v2);
1245     boolean r1b = sb.contains(v1);
1246
1247     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
        sb..size)" */
1248 }
1249
1250 static void contains_add_pre_s_39(ListSet sa, ListSet sb, Object v1, Object v2)
1251 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
1252     sa..contents = sb..contents & sa..size = sb..size"
1253     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1254     ensures "True" */
1255 {
1256     /*: assume "v1 ~= v2 | v1 : sa..contents" */
1257     boolean r1a = sa.contains(v1);
1258     sa.add(v2);
1259
1260     sb.add(v2);
1261     boolean r1b = sb.contains(v1);
1262
1263     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1264 }
1265
1266 static void contains_add_pre_c_39(ListSet sa, ListSet sb, Object v1, Object v2)
1267 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
1268     sa..contents = sb..contents & sa..size = sb..size"
1269     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1270     ensures "True" */
1271 {
1272     /*: assume "~(v1 ~= v2 | v1 : sa..contents)" */
1273     boolean r1a = sa.contains(v1);
1274     sa.add(v2);
1275
1276     sb.add(v2);
1277     boolean r1b = sb.contains(v1);
1278
1279     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1280 }
1281
1282 static void contains_add_between_s_40(ListSet sa, ListSet sb, Object v1, Object v2)
1283 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
1284     sa..contents = sb..contents & sa..size = sb..size"

```

```

1285     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1286     ensures "True" */
1287 {
1288     boolean r1a = sa.contains(v1);
1289     /*: assume "v1 ~= v2 | r1a" */
1290     sa.add(v2);
1291
1292     sb.add(v2);
1293     boolean r1b = sb.contains(v1);
1294
1295     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1296 }
1297
1298 static void contains_add_between_c_40(ListSet sa, ListSet sb, Object v1, Object v2)
1299 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
1300     sa..contents = sb..contents & sa..size = sb..size"
1301     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1302     ensures "True" */
1303 {
1304     boolean r1a = sa.contains(v1);
1305     /*: assume "~(v1 ~= v2 | r1a)" */
1306     sa.add(v2);
1307
1308     sb.add(v2);
1309     boolean r1b = sb.contains(v1);
1310
1311     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1312 }
1313
1314 static void contains_add_post_s_41(ListSet sa, ListSet sb, Object v1, Object v2)
1315 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
1316     sa..contents = sb..contents & sa..size = sb..size"
1317     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1318     ensures "True" */
1319 {
1320     boolean r1a = sa.contains(v1);
1321     sa.add(v2);
1322     /*: assume "v1 ~= v2 | r1a" */
1323
1324     sb.add(v2);
1325     boolean r1b = sb.contains(v1);
1326
1327     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1328 }
1329
1330 static void contains_add_post_c_41(ListSet sa, ListSet sb, Object v1, Object v2)
1331 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
1332     sa..contents = sb..contents & sa..size = sb..size"
1333     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1334     ensures "True" */
1335 {
1336     boolean r1a = sa.contains(v1);
1337     sa.add(v2);
1338     /*: assume "~(v1 ~= v2 | r1a)" */
1339
1340     sb.add(v2);
1341     boolean r1b = sb.contains(v1);
1342
1343     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1344 }
1345
1346 static void contains_contains_pre_s_42(ListSet sa, ListSet sb, Object v1, Object v2)
1347 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
1348     sa..contents = sb..contents & sa..size = sb..size"
1349     ensures "True" */

```

```

1350 {
1351     /*: assume "True" */
1352     boolean r1a = sa.contains(v1);
1353     boolean r2a = sa.contains(v2);
1354
1355     boolean r2b = sb.contains(v2);
1356     boolean r1b = sb.contains(v1);
1357
1358     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1359         sb..size" */
1360 }
1361
1362 static void contains_contains_pre_c_42(ListSet sa, ListSet sb, Object v1, Object v2)
1363 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
1364     sa..contents = sb..contents & sa..size = sb..size"
1365     ensures "True" */
1366 {
1367     /*: assume "~(True)" */
1368     boolean r1a = sa.contains(v1);
1369     boolean r2a = sa.contains(v2);
1370
1371     boolean r2b = sb.contains(v2);
1372     boolean r1b = sb.contains(v1);
1373
1374     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1375         sb..size)" */
1376 }
1377
1378 static void contains_contains_between_s_43(ListSet sa, ListSet sb, Object v1,
1379     Object v2)
1380 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
1381     sa..contents = sb..contents & sa..size = sb..size"
1382     ensures "True" */
1383 {
1384     boolean r1a = sa.contains(v1);
1385     /*: assume "True" */
1386     boolean r2a = sa.contains(v2);
1387
1388     boolean r2b = sb.contains(v2);
1389     boolean r1b = sb.contains(v1);
1390
1391     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1392         sb..size" */
1393 }
1394
1395 static void contains_contains_between_c_43(ListSet sa, ListSet sb, Object v1,
1396     Object v2)
1397 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
1398     sa..contents = sb..contents & sa..size = sb..size"
1399     ensures "True" */
1400 {
1401     boolean r1a = sa.contains(v1);
1402     /*: assume "~(True)" */
1403     boolean r2a = sa.contains(v2);
1404
1405     boolean r2b = sb.contains(v2);
1406     boolean r1b = sb.contains(v1);
1407
1408     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1409         sb..size)" */
1410 }
1411
1412 static void contains_contains_post_s_44(ListSet sa, ListSet sb, Object v1, Object
1413     v2)
1414 /*: requires "sa ~= null & sb ~= null & sa ~= sb &

```

```

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 =
1419         sb..size" */
1420 }
1421
1422 static void contains_contains_post_c_44(ListSet sa, ListSet sb, Object v1, Object
1423     v2)
1424 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
1425     sa..contents = sb..contents & sa..size = sb..size"
1426     ensures "True" */
1427 {
1428     boolean r1a = sa.contains(v1);
1429     boolean r2a = sa.contains(v2);
1430     /*: assume "~(True)" */
1431
1432     boolean r2b = sb.contains(v2);
1433     boolean r1b = sb.contains(v1);
1434
1435     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1436         sb..size)" */
1437 }
1438
1439 static void contains_remove_pre_s_45(ListSet sa, ListSet sb, Object v1, Object v2)
1440 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
1441     sa..contents = sb..contents & sa..size = sb..size"
1442     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1443     ensures "True" */
1444 {
1445     /*: assume "v1 ~= v2 | v1 ~: sa..contents" */
1446     boolean r1a = sa.contains(v1);
1447     boolean r2a = sa.remove(v2);
1448
1449     boolean r2b = sb.remove(v2);
1450     boolean r1b = sb.contains(v1);
1451
1452     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1453         sb..size" */
1454 }
1455
1456 static void contains_remove_pre_c_45(ListSet sa, ListSet sb, Object v1, Object v2)
1457 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
1458     sa..contents = sb..contents & sa..size = sb..size"
1459     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1460     ensures "True" */
1461 {
1462     /*: assume "~(v1 ~= v2 | v1 ~: sa..contents)" */
1463     boolean r1a = sa.contains(v1);
1464     boolean r2a = sa.remove(v2);
1465
1466     boolean r2b = sb.remove(v2);
1467     boolean r1b = sb.contains(v1);
1468
1469     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1470         sb..size)" */
1471 }

```

```

1468 static void contains_remove_between_s_46(ListSet sa, ListSet sb, Object v1, Object
1469 v2)
1470 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
1471 sa..contents = sb..contents & sa..size = sb..size"
1472 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1473 ensures "True" */
1474 {
1475     boolean r1a = sa.contains(v1);
1476     /*: assume "v1 ~= v2 | ~r1a" */
1477     boolean r2a = sa.remove(v2);
1478
1479     boolean r2b = sb.remove(v2);
1480     boolean r1b = sb.contains(v1);
1481
1482     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1483 sb..size" */
1484 }
1485
1486 static void contains_remove_between_c_46(ListSet sa, ListSet sb, Object v1, Object
1487 v2)
1488 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
1489 sa..contents = sb..contents & sa..size = sb..size"
1490 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1491 ensures "True" */
1492 {
1493     boolean r1a = sa.contains(v1);
1494     /*: assume "~(v1 ~= v2 | ~r1a)" */
1495     boolean r2a = sa.remove(v2);
1496
1497     boolean r2b = sb.remove(v2);
1498     boolean r1b = sb.contains(v1);
1499
1500     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1501 sb..size)" */
1502 }
1503
1504 static void contains_remove_post_s_47(ListSet sa, ListSet sb, Object v1, Object v2)
1505 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
1506 sa..contents = sb..contents & sa..size = sb..size"
1507 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1508 ensures "True" */
1509 {
1510     boolean r1a = sa.contains(v1);
1511     boolean r2a = sa.remove(v2);
1512     /*: assume "v1 ~= v2 | ~r1a" */
1513
1514     boolean r2b = sb.remove(v2);
1515     boolean r1b = sb.contains(v1);
1516
1517     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1518 sb..size" */
1519 }
1520
1521 static void contains_remove_post_c_47(ListSet sa, ListSet sb, Object v1, Object v2)
1522 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
1523 sa..contents = sb..contents & sa..size = sb..size"
1524 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1525 ensures "True" */
1526 {
1527     boolean r1a = sa.contains(v1);
1528     boolean r2a = sa.remove(v2);
1529     /*: assume "~(v1 ~= v2 | ~r1a)" */
1530
1531     boolean r2b = sb.remove(v2);
1532     boolean r1b = sb.contains(v1);

```

```

1528     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1529         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
1565     v2)
1566 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
1567     sa..contents = sb..contents & sa..size = sb..size"
1568     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1569     ensures "True" */
1570 {
1571     boolean r1a = sa.contains(v1);
1572     /*: assume "v1 ~= v2 | ~r1a" */
1573     sa.remove(v2);
1574
1575     sb.remove(v2);
1576     boolean r1b = sb.contains(v1);
1577
1578     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1579 }
1580
1581 static void contains_remove_between_c_49(ListSet sa, ListSet sb, Object v1, Object
1582     v2)
1583 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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     boolean r1a = sa.contains(v1);
1589     /*: assume "~(v1 ~= v2 | ~r1a)" */
1590     sa.remove(v2);

```

```

1590     sb.remove(v2);
1591     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(ListSet sa, ListSet sb, Object v1, Object v2)
1597 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(ListSet sa, ListSet sb, Object v1, Object v2)
1613 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(ListSet sa, ListSet sb, Object v1)
1629 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
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 =
1641         sb..size" */
1642 }
1643
1644 static void contains_size_pre_c_51(ListSet sa, ListSet sb, Object v1)
1645 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
1646     sa..contents = sb..contents & sa..size = sb..size"
1647     ensures "True" */
1648 {
1649     /*: assume "~(True)" */
1650     boolean r1a = sa.contains(v1);
1651     int r2a = sa.size();
1652
1653     int r2b = sb.size();
1654     boolean r1b = sb.contains(v1);

```



```

1654
1655     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1656         sb..size)" */
1657 }
1658
1659 static void contains_size_between_s_52(ListSet sa, ListSet sb, Object v1)
1660 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
1661     sa..contents = sb..contents & sa..size = sb..size"
1662     ensures "True" */
1663 {
1664     boolean r1a = sa.contains(v1);
1665     /*: assume "True" */
1666     int r2a = sa.size();
1667
1668     int r2b = sb.size();
1669     boolean r1b = sb.contains(v1);
1670
1671     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1672         sb..size" */
1673 }
1674
1675 static void contains_size_between_c_52(ListSet sa, ListSet sb, Object v1)
1676 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
1677     sa..contents = sb..contents & sa..size = sb..size"
1678     ensures "True" */
1679 {
1680     boolean r1a = sa.contains(v1);
1681     /*: assume "~(True)" */
1682     int r2a = sa.size();
1683
1684     int r2b = sb.size();
1685     boolean r1b = sb.contains(v1);
1686
1687     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1688         sb..size)" */
1689 }
1690
1691 static void contains_size_post_s_53(ListSet sa, ListSet sb, Object v1)
1692 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
1693     sa..contents = sb..contents & sa..size = sb..size"
1694     ensures "True" */
1695 {
1696     boolean r1a = sa.contains(v1);
1697     int r2a = sa.size();
1698     /*: assume "True" */
1699
1700     int r2b = sb.size();
1701     boolean r1b = sb.contains(v1);
1702
1703     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1704         sb..size" */
1705 }
1706
1707 static void contains_size_post_c_53(ListSet sa, ListSet sb, Object v1)
1708 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
1709     sa..contents = sb..contents & sa..size = sb..size"
1710     ensures "True" */
1711 {
1712     boolean r1a = sa.contains(v1);
1713     int r2a = sa.size();
1714     /*: assume "~(True)" */
1715
1716     int r2b = sb.size();
1717     boolean r1b = sb.contains(v1);
1718

```

```

1715     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1716         sb..size)" */
1717 }
1718 static void remove_add_pre_s_54(ListSet sa, ListSet sb, Object v1, Object v2)
1719 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
1720     sa..contents = sb..contents & sa..size = sb..size"
1721     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1722     ensures "True" */
1723 {
1724     /*: assume "v1 ~= v2" */
1725     boolean r1a = sa.remove(v1);
1726     boolean r2a = sa.add(v2);
1727
1728     boolean r2b = sb.add(v2);
1729     boolean r1b = sb.remove(v1);
1730
1731     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1732         sb..size" */
1733 }
1734 static void remove_add_pre_c_54(ListSet sa, ListSet sb, Object v1, Object v2)
1735 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
1736     sa..contents = sb..contents & sa..size = sb..size"
1737     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1738     ensures "True" */
1739 {
1740     /*: assume "~(v1 ~= v2)" */
1741     boolean r1a = sa.remove(v1);
1742     boolean r2a = sa.add(v2);
1743
1744     boolean r2b = sb.add(v2);
1745     boolean r1b = sb.remove(v1);
1746
1747     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1748         sb..size)" */
1749 }
1750 static void remove_add_between_s_55(ListSet sa, ListSet sb, Object v1, Object v2)
1751 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
1752     sa..contents = sb..contents & sa..size = sb..size"
1753     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1754     ensures "True" */
1755 {
1756     boolean r1a = sa.remove(v1);
1757     /*: assume "v1 ~= v2" */
1758     boolean r2a = sa.add(v2);
1759
1760     boolean r2b = sb.add(v2);
1761     boolean r1b = sb.remove(v1);
1762
1763     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1764         sb..size" */
1765 }
1766 static void remove_add_between_c_55(ListSet sa, ListSet sb, Object v1, Object v2)
1767 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
1768     sa..contents = sb..contents & sa..size = sb..size"
1769     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1770     ensures "True" */
1771 {
1772     boolean r1a = sa.remove(v1);
1773     /*: assume "~(v1 ~= v2)" */
1774     boolean r2a = sa.add(v2);
1775

```

```

1776     boolean r2b = sb.add(v2);
1777     boolean r1b = sb.remove(v1);
1778
1779     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1780         sb..size)" */
1781 }
1782
1783 static void remove_add_post_s_56(ListSet sa, ListSet sb, Object v1, Object v2)
1784 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
1785     sa..contents = sb..contents & sa..size = sb..size"
1786     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1787     ensures "True" */
1788 {
1789     boolean r1a = sa.remove(v1);
1790     boolean r2a = sa.add(v2);
1791     /*: assume "v1 ~= v2" */
1792
1793     boolean r2b = sb.add(v2);
1794     boolean r1b = sb.remove(v1);
1795
1796     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1797         sb..size" */
1798 }
1799
1800 static void remove_add_post_c_56(ListSet sa, ListSet sb, Object v1, Object v2)
1801 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
1802     sa..contents = sb..contents & sa..size = sb..size"
1803     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1804     ensures "True" */
1805 {
1806     boolean r1a = sa.remove(v1);
1807     boolean r2a = sa.add(v2);
1808     /*: assume "~(v1 ~= v2)" */
1809
1810     boolean r2b = sb.add(v2);
1811     boolean r1b = sb.remove(v1);
1812
1813     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1814         sb..size)" */
1815 }
1816
1817 static void remove_add_pre_s_57(ListSet sa, ListSet sb, Object v1, Object v2)
1818 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
1819     sa..contents = sb..contents & sa..size = sb..size"
1820     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1821     ensures "True" */
1822 {
1823     /*: assume "v1 ~= v2" */
1824     boolean r1a = sa.remove(v1);
1825     sa.add(v2);
1826
1827     sb.add(v2);
1828     boolean r1b = sb.remove(v1);
1829
1830     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1831 }
1832
1833 static void remove_add_pre_c_57(ListSet sa, ListSet sb, Object v1, Object v2)
1834 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
1835     sa..contents = sb..contents & sa..size = sb..size"
1836     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1837     ensures "True" */
1838 {
1839     /*: assume "~(v1 ~= v2)" */
1840     boolean r1a = sa.remove(v1);

```

```

1838     sa.add(v2);
1839
1840     sb.add(v2);
1841     boolean r1b = sb.remove(v1);
1842
1843     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1844 }
1845
1846 static void remove_add_between_s_58(ListSet sa, ListSet sb, Object v1, Object v2)
1847 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
1848     sa..contents = sb..contents & sa..size = sb..size"
1849 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1850 ensures "True" */
1851 {
1852     boolean r1a = sa.remove(v1);
1853     /*: assume "v1 ~= v2" */
1854     sa.add(v2);
1855
1856     sb.add(v2);
1857     boolean r1b = sb.remove(v1);
1858
1859     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1860 }
1861
1862 static void remove_add_between_c_58(ListSet sa, ListSet sb, Object v1, Object v2)
1863 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
1864     sa..contents = sb..contents & sa..size = sb..size"
1865 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1866 ensures "True" */
1867 {
1868     boolean r1a = sa.remove(v1);
1869     /*: assume "~(v1 ~= v2)" */
1870     sa.add(v2);
1871
1872     sb.add(v2);
1873     boolean r1b = sb.remove(v1);
1874
1875     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1876 }
1877
1878 static void remove_add_post_s_59(ListSet sa, ListSet sb, Object v1, Object v2)
1879 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
1880     sa..contents = sb..contents & sa..size = sb..size"
1881 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1882 ensures "True" */
1883 {
1884     boolean r1a = sa.remove(v1);
1885     sa.add(v2);
1886     /*: assume "v1 ~= v2" */
1887
1888     sb.add(v2);
1889     boolean r1b = sb.remove(v1);
1890
1891     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1892 }
1893
1894 static void remove_add_post_c_59(ListSet sa, ListSet sb, Object v1, Object v2)
1895 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
1896     sa..contents = sb..contents & sa..size = sb..size"
1897 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1898 ensures "True" */
1899 {
1900     boolean r1a = sa.remove(v1);
1901     sa.add(v2);
1902     /*: assume "~(v1 ~= v2)" */

```

```

1903     sb.add(v2);
1904     boolean r1b = sb.remove(v1);
1905
1906     /*: assert "~(ria = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1907 }
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 "ria = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1924     sb..size" */
1925 }
1926
1927 static void remove_contains_pre_c_60(ListSet sa, ListSet sb, Object v1, Object v2)
1928 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
1929     sa..contents = sb..contents & sa..size = sb..size"
1930 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1931 ensures "True" */
1932 {
1933     /*: assume "~(v1 ~= v2 | v1 ~: sa..contents)" */
1934     boolean r1a = sa.remove(v1);
1935     boolean r2a = sa.contains(v2);
1936
1937     boolean r2b = sb.contains(v2);
1938     boolean r1b = sb.remove(v1);
1939
1940     /*: assert "~(ria = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1941     sb..size)" */
1942 }
1943
1944 static void remove_contains_between_s_61(ListSet sa, ListSet sb, Object v1, Object
1945     v2)
1946 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
1947     sa..contents = sb..contents & sa..size = sb..size"
1948 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1949 ensures "True" */
1950 {
1951     boolean r1a = sa.remove(v1);
1952     /*: assume "v1 ~= v2 | ~r1a" */
1953     boolean r2a = sa.contains(v2);
1954
1955     boolean r2b = sb.contains(v2);
1956     boolean r1b = sb.remove(v1);
1957
1958     /*: assert "ria = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1959     sb..size" */
1960 }
1961
1962 static void remove_contains_between_c_61(ListSet sa, ListSet sb, Object v1, Object
1963     v2)
1964 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
1965     sa..contents = sb..contents & sa..size = sb..size"
1966 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1967 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 =
1972         sb..size)" */
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 =
1988         sb..size" */
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 =
2004         sb..size)" */
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 =
2020         sb..size" */
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 &

```

```

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(ListSet sa, ListSet sb, Object v1, Object v2)
2039 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
        sa..contents = sb..contents & sa..size = sb..size"
2040 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2041 ensures "True" */
2042 {
2043     boolean r1a = sa.remove(v1);
2044     /*: assume "v1 ~= v2 | ~r1a" */
2045     boolean r2a = sa.remove(v2);
2046
2047     boolean r2b = sb.remove(v2);
2048     boolean r1b = sb.remove(v1);
2049
2050     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
        sb..size" */
2051 }
2052
2053
2054 static void remove_remove_between_c_64(ListSet sa, ListSet sb, Object v1, Object v2)
2055 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
        sa..contents = sb..contents & sa..size = sb..size"
2056 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2057 ensures "True" */
2058 {
2059     boolean r1a = sa.remove(v1);
2060     /*: assume "~(v1 ~= v2 | ~r1a)" */
2061     boolean r2a = sa.remove(v2);
2062
2063     boolean r2b = sb.remove(v2);
2064     boolean r1b = sb.remove(v1);
2065
2066     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
        sb..size)" */
2067 }
2068
2069
2070 static void remove_remove_post_s_65(ListSet sa, ListSet sb, Object v1, Object v2)
2071 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
        sa..contents = sb..contents & sa..size = sb..size"
2072 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2073 ensures "True" */
2074 {
2075     boolean r1a = sa.remove(v1);
2076     boolean r2a = sa.remove(v2);
2077     /*: assume "v1 ~= v2 | ~r1a" */
2078
2079     boolean r2b = sb.remove(v2);
2080     boolean r1b = sb.remove(v1);
2081
2082     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
        sb..size" */
2083 }
2084

```

```

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 =
2212      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 =
2228      sb..size)" */
2229 }
2230
2231 static void remove_size_between_s_70(ListSet sa, ListSet sb, Object v1)
2232 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
2233      sa..contents = sb..contents & sa..size = sb..size"
2234      modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2235      ensures "True" */
2236 {
2237     boolean r1a = sa.remove(v1);
2238     /*: assume "~r1a" */
2239     int r2a = sa.size();
2240
2241     int r2b = sb.size();
2242     boolean r1b = sb.remove(v1);
2243
2244     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2245      sb..size" */
2246 }
2247
2248 static void remove_size_between_c_70(ListSet sa, ListSet sb, Object v1)
2249 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
2250      sa..contents = sb..contents & sa..size = sb..size"
2251      modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2252      ensures "True" */
2253 {
2254     boolean r1a = sa.remove(v1);
2255     /*: assume "~(r1a)" */
2256     int r2a = sa.size();
2257
2258     int r2b = sb.size();
2259     boolean r1b = sb.remove(v1);
2260
2261     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2262      sb..size)" */
2263 }
2264
2265 static void remove_size_post_s_71(ListSet sa, ListSet sb, Object v1)
2266 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
2267      sa..contents = sb..contents & sa..size = sb..size"
2268      modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2269      ensures "True" */
2270 {
2271     boolean r1a = sa.remove(v1);
2272     int r2a = sa.size();
2273     /*: assume "~r1a" */
2274
2275     int r2b = sb.size();
2276     boolean r1b = sb.remove(v1);

```

```

2275     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2276         sb..size" */
2277 }
2278 static void remove_size_post_c_71(ListSet sa, ListSet sb, Object v1)
2279 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
2280     sa..contents = sb..contents & sa..size = sb..size"
2281     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2282     ensures "True" */
2283 {
2284     boolean r1a = sa.remove(v1);
2285     int r2a = sa.size();
2286     /*: assume "~(r1a)" */
2287
2288     int r2b = sb.size();
2289     boolean r1b = sb.remove(v1);
2290
2291     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2292         sb..size)" */
2293 }
2294 static void remove_add_pre_s_72(ListSet sa, ListSet sb, Object v1, Object v2)
2295 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
2296     sa..contents = sb..contents & sa..size = sb..size"
2297     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2298     ensures "True" */
2299 {
2300     /*: assume "v1 ~= v2" */
2301     sa.remove(v1);
2302     boolean r2a = sa.add(v2);
2303
2304     boolean r2b = sb.add(v2);
2305     sb.remove(v1);
2306
2307     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2308 }
2309 static void remove_add_pre_c_72(ListSet sa, ListSet sb, Object v1, Object v2)
2310 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
2311     sa..contents = sb..contents & sa..size = sb..size"
2312     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2313     ensures "True" */
2314 {
2315     /*: assume "~(v1 ~= v2)" */
2316     sa.remove(v1);
2317     boolean r2a = sa.add(v2);
2318
2319     boolean r2b = sb.add(v2);
2320     sb.remove(v1);
2321
2322     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2323 }
2324 static void remove_add_between_s_73(ListSet sa, ListSet sb, Object v1, Object v2)
2325 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
2326     sa..contents = sb..contents & sa..size = sb..size"
2327     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2328     ensures "True" */
2329 {
2330     sa.remove(v1);
2331     /*: assume "v1 ~= v2" */
2332     boolean r2a = sa.add(v2);
2333
2334     boolean r2b = sb.add(v2);
2335     sb.remove(v1);
2336
2337

```

```

2338     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2339 }
2340
2341
2342 static void remove_add_between_c_73(ListSet sa, ListSet sb, Object v1, Object v2)
2343 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
2344     sa..contents = sb..contents & sa..size = sb..size"
2345     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2346     ensures "True" */
2347 {
2348     sa.remove(v1);
2349     /*: assume "~(v1 ~= v2)" */
2350     boolean r2a = sa.add(v2);
2351
2352     boolean r2b = sb.add(v2);
2353     sb.remove(v1);
2354
2355     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2356 }
2357
2358 static void remove_add_post_s_74(ListSet sa, ListSet sb, Object v1, Object v2)
2359 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(ListSet sa, ListSet sb, Object v1, Object v2)
2375 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(ListSet sa, ListSet sb, Object v1, Object v2)
2391 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(ListSet sa, ListSet sb, Object v1, Object v2)
2407 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(ListSet sa, ListSet sb, Object v1, Object v2)
2423 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(ListSet sa, ListSet sb, Object v1, Object v2)
2439 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(ListSet sa, ListSet sb, Object v1, Object v2)
2455 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(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
2519     v2)
2520 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
2521           sa..contents = sb..contents & sa..size = sb..size"
2522   modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2523   ensures "True" */
2524 {
2525     sa.remove(v1);
2526     /*: assume "v1 ~= v2 | v1 ~: sa..(old contents)" */
2527     boolean r2a = sa.contains(v2);
2528
2529     boolean r2b = sb.contains(v2);
2530     sb.remove(v1);
2531
2532     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */

```

```

2532 }
2533
2534 static void remove_contains_between_c_79(ListSet sa, ListSet sb, Object v1, Object
      v2)
2535 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
2536             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(ListSet sa, ListSet sb, Object v1, Object v2)
2551 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= 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(ListSet sa, ListSet sb, Object v1, Object v2)
2567 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= 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(ListSet sa, ListSet sb, Object v1, Object v2)
2583 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(ListSet sa, ListSet sb, Object v1, Object v2)
2599 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
2600      sa..contents = sb..contents & sa..size = sb..size"
2601      modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2602      ensures "True" */
2603 {
2604     /*: assume "~(v1 ~= v2 | v1 ~: sa..contents)" */
2605     sa.remove(v1);
2606     boolean r2a = sa.remove(v2);
2607
2608     boolean r2b = sb.remove(v2);
2609     sb.remove(v1);
2610
2611     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2612 }
2613
2614 static void remove_remove_between_s_82(ListSet sa, ListSet sb, Object v1, Object v2)
2615 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
2616      sa..contents = sb..contents & sa..size = sb..size"
2617      modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2618      ensures "True" */
2619 {
2620     sa.remove(v1);
2621     /*: assume "v1 ~= v2 | v1 ~: sa..(old contents)" */
2622     boolean r2a = sa.remove(v2);
2623
2624     boolean r2b = sb.remove(v2);
2625     sb.remove(v1);
2626
2627     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2628 }
2629
2630 static void remove_remove_between_c_82(ListSet sa, ListSet sb, Object v1, Object v2)
2631 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
2632      sa..contents = sb..contents & sa..size = sb..size"
2633      modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2634      ensures "True" */
2635 {
2636     sa.remove(v1);
2637     /*: assume "~(v1 ~= v2 | v1 ~: sa..(old contents))" */
2638     boolean r2a = sa.remove(v2);
2639
2640     boolean r2b = sb.remove(v2);
2641     sb.remove(v1);
2642
2643     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2644 }
2645
2646 static void remove_remove_post_s_83(ListSet sa, ListSet sb, Object v1, Object v2)
2647 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
2648      sa..contents = sb..contents & sa..size = sb..size"
2649      modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2650      ensures "True" */
2651 {
2652     sa.remove(v1);
2653     boolean r2a = sa.remove(v2);
2654     /*: assume "v1 ~= v2 | v1 ~: sa..(old contents)" */
2655
2656     boolean r2b = sb.remove(v2);
2657     sb.remove(v1);
2658
2659     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2660 }

```



```

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

```

```

2726 static void remove_remove_between_c_85(ListSet sa, ListSet sb, Object v1, Object v2)
2727 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
2728     sa..contents = sb..contents & sa..size = sb..size"
2729 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2730 ensures "True" */
2731 {
2732     sa.remove(v1);
2733     /*: assume "~(True)" */
2734     sa.remove(v2);
2735
2736     sb.remove(v2);
2737     sb.remove(v1);
2738
2739     /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
2740 }
2741
2742 static void remove_remove_post_s_86(ListSet sa, ListSet sb, Object v1, Object v2)
2743 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
2744     sa..contents = sb..contents & sa..size = sb..size"
2745 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2746 ensures "True" */
2747 {
2748     sa.remove(v1);
2749     sa.remove(v2);
2750     /*: assume "True" */
2751
2752     sb.remove(v2);
2753     sb.remove(v1);
2754
2755     /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
2756 }
2757
2758 static void remove_remove_post_c_86(ListSet sa, ListSet sb, Object v1, Object v2)
2759 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null & v2 ~= null &
2760     sa..contents = sb..contents & sa..size = sb..size"
2761 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2762 ensures "True" */
2763 {
2764     sa.remove(v1);
2765     sa.remove(v2);
2766     /*: assume "~(True)" */
2767
2768     sb.remove(v2);
2769     sb.remove(v1);
2770
2771     /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
2772 }
2773
2774 static void remove_size_pre_s_87(ListSet sa, ListSet sb, Object v1)
2775 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v1 ~= null &
2776     sa..contents = sb..contents & sa..size = sb..size"
2777 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2778 ensures "True" */
2779 {
2780     /*: assume "v1 ~: sa..contents" */
2781     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(ListSet sa, ListSet sb, Object v1)

```

```

2791  /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(ListSet sa, ListSet sb, Object v1)
2807  /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(ListSet sa, ListSet sb, Object v1)
2823  /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(ListSet sa, ListSet sb, Object v1)
2839  /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(ListSet sa, ListSet sb, Object v1)
2855  /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(ListSet sa, ListSet sb, Object v2)
2871 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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 =
2884         sb..size" */
2885 }
2886
2887 static void size_add_pre_c_90(ListSet sa, ListSet sb, Object v2)
2888 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
2889     sa..contents = sb..contents & sa..size = sb..size"
2890     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2891     ensures "True" */
2892 {
2893     /*: assume "~(v2 : sa..contents)" */
2894     int r1a = sa.size();
2895     boolean r2a = sa.add(v2);
2896
2897     boolean r2b = sb.add(v2);
2898     int r1b = sb.size();
2899
2900     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2901         sb..size)" */
2902 }
2903
2904 static void size_add_between_s_91(ListSet sa, ListSet sb, Object v2)
2905 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
2906     sa..contents = sb..contents & sa..size = sb..size"
2907     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2908     ensures "True" */
2909 {
2910     int r1a = sa.size();
2911     /*: assume "v2 : sa..contents" */
2912     boolean r2a = sa.add(v2);
2913
2914     boolean r2b = sb.add(v2);
2915     int r1b = sb.size();
2916
2917     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2918         sb..size" */
2919 }

```

```

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 =
2932      sb..size)" */
2933 }
2934
2935 static void size_add_post_s_92(ListSet sa, ListSet sb, Object v2)
2936 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
2937      sa..contents = sb..contents & sa..size = sb..size"
2938      modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2939      ensures "True" */
2940 {
2941     int r1a = sa.size();
2942     boolean r2a = sa.add(v2);
2943     /*: assume "~r2a" */
2944
2945     boolean r2b = sb.add(v2);
2946     int r1b = sb.size();
2947
2948     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2949      sb..size" */
2950 }
2951
2952 static void size_add_post_c_92(ListSet sa, ListSet sb, Object v2)
2953 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
2954      sa..contents = sb..contents & sa..size = sb..size"
2955      modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2956      ensures "True" */
2957 {
2958     int r1a = sa.size();
2959     boolean r2a = sa.add(v2);
2960     /*: assume "~(~r2a)" */
2961
2962     boolean r2b = sb.add(v2);
2963     int r1b = sb.size();
2964
2965     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2966      sb..size)" */
2967 }
2968
2969 static void size_add_pre_s_93(ListSet sa, ListSet sb, Object v2)
2970 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
2971      sa..contents = sb..contents & sa..size = sb..size"
2972      modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2973      ensures "True" */
2974 {
2975     /*: assume "v2 : sa..contents" */
2976     int r1a = sa.size();
2977     sa.add(v2);
2978
2979     sb.add(v2);
2980     int r1b = sb.size();
2981
2982     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */

```

```

2980 }
2981
2982 static void size_add_pre_c_93(ListSet sa, ListSet sb, Object v2)
2983 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
2984     sa..contents = sb..contents & sa..size = sb..size"
2985     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2986     ensures "True" */
2987 {
2988     /*: assume "~(v2 : sa..contents)" */
2989     int r1a = sa.size();
2990     sa.add(v2);
2991
2992     sb.add(v2);
2993     int r1b = sb.size();
2994
2995     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
2996 }
2997
2998 static void size_add_between_s_94(ListSet sa, ListSet sb, Object v2)
2999 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
3000     sa..contents = sb..contents & sa..size = sb..size"
3001     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3002     ensures "True" */
3003 {
3004     int r1a = sa.size();
3005     /*: assume "v2 : sa..contents" */
3006     sa.add(v2);
3007
3008     sb.add(v2);
3009     int r1b = sb.size();
3010
3011     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
3012 }
3013
3014 static void size_add_between_c_94(ListSet sa, ListSet sb, Object v2)
3015 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
3016     sa..contents = sb..contents & sa..size = sb..size"
3017     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3018     ensures "True" */
3019 {
3020     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(ListSet sa, ListSet sb, Object v2)
3031 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(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 =
3077           sb..size" */
3078 }
3079
3080 static void size_contains_pre_c_96(ListSet sa, ListSet sb, Object v2)
3081 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
3082           sa..contents = sb..contents & sa..size = sb..size"
3083   ensures "True" */
3084 {
3085   /*: assume "~(True)" */
3086   int r1a = sa.size();
3087   boolean r2a = sa.contains(v2);
3088
3089   boolean r2b = sb.contains(v2);
3090   int r1b = sb.size();
3091
3092   /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
3093           sb..size)" */
3094 }
3095
3096 static void size_contains_between_s_97(ListSet sa, ListSet sb, Object v2)
3097 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
3098           sa..contents = sb..contents & sa..size = sb..size"
3099   ensures "True" */
3100 {
3101   int r1a = sa.size();
3102   /*: assume "True" */
3103   boolean r2a = sa.contains(v2);
3104
3105   boolean r2b = sb.contains(v2);
3106   int r1b = sb.size();
3107
3108   /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
3109           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 =
3122                sb..size)" */
3123     }
3124
3125     static void size_contains_post_s_98(ListSet sa, ListSet sb, Object v2)
3126     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
3127                sa..contents = sb..contents & sa..size = sb..size"
3128        ensures "True" */
3129     {
3130         int r1a = sa.size();
3131         boolean r2a = sa.contains(v2);
3132         /*: assume "True" */
3133
3134         boolean r2b = sb.contains(v2);
3135         int r1b = sb.size();
3136
3137         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
3138                sb..size" */
3139     }
3140
3141     static void size_contains_post_c_98(ListSet sa, ListSet sb, Object v2)
3142     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
3143                sa..contents = sb..contents & sa..size = sb..size"
3144        ensures "True" */
3145     {
3146         int r1a = sa.size();
3147         boolean r2a = sa.contains(v2);
3148         /*: assume "~(True)" */
3149
3150         boolean r2b = sb.contains(v2);
3151         int r1b = sb.size();
3152
3153         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
3154                sb..size)" */
3155     }
3156
3157     static void size_remove_pre_s_99(ListSet sa, ListSet sb, Object v2)
3158     /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
3159                sa..contents = sb..contents & sa..size = sb..size"
3160        modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3161        ensures "True" */
3162     {
3163         /*: assume "v2 ~: sa..contents" */
3164         int r1a = sa.size();
3165         boolean r2a = sa.remove(v2);
3166
3167         boolean r2b = sb.remove(v2);
3168         int r1b = sb.size();
3169
3170         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
3171                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 =
3184           sb..size)" */
3185 }
3186
3187 static void size_remove_between_s_100(ListSet sa, ListSet sb, Object v2)
3188 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
3189           sa..contents = sb..contents & sa..size = sb..size"
3190   modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3191   ensures "True" */
3192 {
3193   int r1a = sa.size();
3194   /*: assume "v2 ~: sa..contents" */
3195   boolean r2a = sa.remove(v2);
3196
3197   boolean r2b = sb.remove(v2);
3198   int r1b = sb.size();
3199
3200   /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
3201           sb..size" */
3202 }
3203
3204 static void size_remove_between_c_100(ListSet sa, ListSet sb, Object v2)
3205 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
3206           sa..contents = sb..contents & sa..size = sb..size"
3207   modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3208   ensures "True" */
3209 {
3210   int r1a = sa.size();
3211   /*: assume "~(v2 ~: sa..contents)" */
3212   boolean r2a = sa.remove(v2);
3213
3214   boolean r2b = sb.remove(v2);
3215   int r1b = sb.size();
3216
3217   /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
3218           sb..size)" */
3219 }
3220
3221 static void size_remove_post_s_101(ListSet sa, ListSet sb, Object v2)
3222 /*: requires "sa ~= null & sb ~= null & sa ~= sb & v2 ~= null &
3223           sa..contents = sb..contents & sa..size = sb..size"
3224   modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3225   ensures "True" */
3226 {
3227   int r1a = sa.size();
3228   boolean r2a = sa.remove(v2);
3229   /*: assume "~r2a" */
3230
3231   boolean r2b = sb.remove(v2);
3232   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();

```

```

3419     /*: assert "ria = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
3420         sb..size" */
3421 }
3422
3423 static void size_size_post_c_107(ListSet sa, ListSet sb)
3424 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
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 "~(ria = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
3436         sb..size)" */
3437 }
3438 }

```

### B.2.3 Inverse Testing Methods

Listing 6. ListSetInv.java

```

1 class ListSetInv {
2     static void add_0(ListSet s, Object v)
3     /*: requires "s ~= null & v ~= null"
4         modifies "s..contents", "s..size"
5         ensures "True" */
6     {
7         boolean r = s.add(v);
8         if (r) { s.remove(v); }
9
10        /*: assert "s..contents = s..(old contents) & s..size = s..(old size)" */
11    }
12
13    static void remove_1(ListSet s, Object v)
14    /*: requires "s ~= null & v ~= null"
15        modifies "s..contents", "s..size"
16        ensures "True" */
17    {
18        boolean r = s.remove(v);
19        if (r) { s.add(v); }
20
21        /*: assert "s..contents = s..(old contents) & s..size = s..(old size)" */
22    }
23 }
24

```

## B.3 HashSet

### B.3.1 Data Structure

Listing 7. HashSet.java

```

1 public /*: claimedby HashSet */ class Node {
2     public Object value;
3     public Node next;
4
5     /*: public ghost specvar conts :: "obj set" = "{}";
6         invariant ConDef: "this ~= null --> conts = {value} Un next..conts & (value ~:
7             next..conts)";
8         invariant ConAlloc: "ALL x. x : conts --> x : alloc";

```

```

8       invariant ConNull: "null..conts = {}"; */
9   }
10
11 public class HashSet {
12     private Node[] table = null;
13     private int _size;
14
15     /*: public ghost specvar init :: "bool" = "False";
16
17     static specvar abs :: "(int => int)"
18     vardefs "abs == (%i1. (if (i1 < 0) then (-i1) else i1))";
19
20     static specvar h :: "(obj => int => int)";
21     vardefs "h == (%o1. (%i1. ((abs (hashFunc o1)) mod i1)))";
22
23     invariant HashInv: "init --> (ALL k. 0 <= (h k (table..length)) & (h k
24         (table..length)) < table..length)";
25
26     public ghost specvar contents :: "obj set" = "{}";
27     invariant ContentsDefInv: "init --> contents = {v. v : table.[(h v
28         (table..length))].conts}";
29
30     invariant Coherence: "init --> (ALL i v. 0 <= i & i < table..length --> v :
31         table.[i].conts --> h v (table..length) = i)";
32
33     invariant TableNotNull: "init --> table ~= null";
34
35     invariant TableHidden: "init --> table : hidden";
36     invariant NodeHidden1: "init --> (ALL i. 0 <= i & i < table..length & table.[i]
37         ~= null --> table.[i] : hidden)";
38     invariant NodeHidden2: "ALL n. n : Node & n : alloc & n ~= null & n..next ~=
39         null --> n..next : hidden";
40
41     invariant FirstInjInv: "init --> (ALL i x y. y = x..next & y ~= null & 0 <= i &
42         i < table..length --> y ~= table.[i])";
43     invariant NextInjInv: "ALL x1 x2 y. y ~= null & y = x1..next & y = x2..next -->
44         x1 = x2";
45     invariant ElementInjInv: "init --> (ALL hs i j. hs : HashSet & hs : alloc &
46         hs..init & 0 <= i & i < hs..table..length & 0 <= j & j < table..length &
47         hs..table.[i] = table.[j] & table.[j] ~= null --> hs = this & i = j)";
48     invariant TableInjInv: "ALL hs. hs..table = table & table ~= null --> hs =
49         this";
50
51     public specvar size :: "int"
52     vardefs "size == _size"
53     invariant CardInv: "init --> _size = card (contents)" */
54
55 public HashSet()
56 /*: modifies "contents", "size", "init"
57 ensures "init & contents = {} & size = 0" */
58 {
59     table = new /*: hidden */ Node[11];
60     _size = 0;
61
62     /*: "contents" := "{}";
63     "init" := "True"; */
64
65     /*: note NewNotHT: "table ~: HashSet"; */
66     {
67         /*: localize;
68         note ElemInj1: "ALL hs1 i j. hs1 : HashSet & hs1 : alloc & hs1..init &
69             0 <= i & i < hs1..table..length & 0 <= j & j < table..length &
70             hs1..table.[i] = table.[j] & hs1..table.[i] ~= null --> hs1 = this &
71             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 =
60     j";
    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,
61     NewNotHT;
    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)" */
165         (curr != null) {
166         if (curr.value == v)
167             return true;
168         curr = curr.next;
169     }
170     return false;
171 }
172
173 public boolean remove(Object v)
174 /*: requires "init & v ~= null"
175    modifies "contents", "size"
176    ensures "(v : old contents --> contents = old contents - {v} & size = old size
177            - 1 & result) & (v ~: old contents --> contents = old contents & size = old
178            size & ~result)" */
179 {
180     if (_contains(v)) {
181         _remove(v);
182         return true;
183     } else {
184         return false;
185     }
186 }
187
188 private void removeFirst(Object v, int hc)
189 /*: requires "init & v ~= null & comment ''InContents'' (v : contents) & comment
190    ''KFound'' (v = table.[hc]..value) & comment ''HCProps'' (0 <= hc & hc <
191    table..length & hc = h v (table..length)) & theinvs"
192    modifies "contents", "size", "conts", "next", "arrayState", "_size"
193    ensures "(contents = old contents - {v}) & (size = old size - 1) & comment
194    ''C3'' (ALL a i. a ~= table --> a.[i] = old (a.[i])) & theinvs" */
195 {
196     Node f = table[hc];
197     Node second = f.next;
198
199     f.next = null;
200     /*: "f..conts" := "{f..value}"; */
201
202     table[hc] = second;
203     /*: "contents" := "old contents - {v}"; */
204
205     _size = _size - 1;
206
207     /*: note ThisProps: "this : alloc & this : HashSet & init";
208        note OldContents: "old contents = {w. w : old (table.[(h w
209        (table..length))]..conts)}" from ContentsDefInv, ThisProps;
210        note FNonNull: "f ~= null";
211        note FProps: "f : Node & f : alloc" from unalloc_lonely, array_pointsto,
212        ThisProps;

```

```

205     note VFound: "v = f..value" from InContents, OldContents, ConDef, KFound,
206         FProps, FNonNull, HCProps;
207     note Acyclic: "fieldRead (old next) f ~= f" from FNonNull, HCProps,
208         FirstInjInv, ThisProps; */
209 {
210     /*: pickAny hs::obj suchThat ContentsDefHyp: "hs : alloc & hs : HashSet &
211         hs..init";
212     note ContentsThis: "hs = this --> hs..contents = {w. w : hs..table.[(h
213         w (hs..table..length))].conts}" from OldContents, ElementInjInv,
214         Acyclic, ThisProps, KFound, VFound, ConDef, FProps, FNonNull,
215         HashInv, HCProps; */
216     {
217         /*: assuming NotThisHyp: "hs ~= this";
218         note OldHTContents: "fieldRead (old HashSet.contents) hs = {w. w :
219             (fieldRead (old conts) (arrayRead (old arrayState) (hs..table)
220             (h w (hs..table..length))))}" from ContentsDefHyp, NotThisHyp,
221             ContentsDefInv;
222         note TableNotEq: "hs..table ~= table";
223         note ContentsOther: "hs..contents = {w. w : hs..table.[(h w
224             (hs..table..length))].conts}" from ContentsDefHyp, NotThisHyp,
225             HashInv, ElementInjInv, HCProps, ThisProps, FNonNull,
226             OldHTContents, TableNotEq; */
227     }
228     /*: cases "hs = this", "hs ~= this" for ContentsCases: "hs..contents = {w.
229         w : hs..table.[(h w (hs..table..length))].conts}" from ContentsThis,
230         ContentsOther;
231     note ContentsDefPostCond: "hs..contents = {w. w : hs..table.[(h w
232         (hs..table..length))].conts}" from ContentCases forSuch hs; */
233 }
234 /*: note CoherencePostCond: "theinv Coherence" from Coherence, Acyclic, ConDef,
235     ConNull, FNonNull, TableInjInv, FProps, HCProps; */
236
237 /*: note ContentsPost: "contents = old contents - {v}" */
238 {
239     /*: pickAny x::obj */
240     {
241         /*: assuming CardHyp: "x : alloc & x : HashSet & x..init" */
242         {
243             /*: note ThisProps: "this : old alloc & this : HashSet & old init"
244                 */
245             /*: note OldCard: "old _size = card (old contents)" from ThisProps,
246                 CardInv */
247             /*: note NewLength: "_size = old _size - 1" */
248             /*: note NewNotInOld: "v : old contents" */
249             /*: note XIsThisCard: "_size = card (contents)" from OldCard,
250                 NewLength, NewNotInOld, ContentsPost */
251         }
252     }
253     {
254         /*: assuming XNotThisHyp: "x ~= this" */
255         /*: note XInOldAlloc: "x : old alloc" */
256         /*: note XOldInit: "x..(old HashSet.init)" */
257         /*: note OldCard: "x..(old HashSet._size) = card (x..(old
258             HashSet.contents))" from CardHyp, XNotThisHyp, XInOldAlloc,
259             XOldInit, CardInv */
260         /*: note XLengthEq: "x.._size = x..(old HashSet._size)" */
261         {
262             /*: localize */
263             /*: note XContentsForw: "ALL y. y : x..contents --> y : x..(old
264                 HashSet.contents)" */
265             /*: note XContentsBack: "ALL y. y : x..(old HashSet.contents)
266                 --> y : x..contents" */
267             /*: note XContentsUnchanged: "x..contents = x..(old
268                 HashSet.contents)" from XContentsForw, XContentsBack */
269         }
270     }
271 }

```

```

245         /*: note XNotThisCard: "x.._size = card (x..contents)" from
246             XLengthEq, OldCard, XContentsUnchanged */
247     }
248     /*: note CardConc: "x.._size = card (x..contents)" from XIsThisCard,
249         XNotThisCard */
250 }
251 }
252
253 private Object removeFromBucket(Object v, int hc)
254 /*: requires "comment ''Init'' init & v ~= null & comment ''InContents'' (v :
255     contents) & comment ''KNotFound'' (v ~= table.[hc]..value) & comment ''HCPProps''
256     (0 <= hc & hc < table..length & hc = h v (table..length)) & theinvs"
257     modifies "contents", "size", "conts", "next", "arrayState", "_size"
258     ensures "(contents = old contents - {v}) & (size = old size - 1) & (ALL a i. a
259         ~= table --> a.[i] = old (a.[i])) & theinvs" */
260 {
261     Node f = table[hc];
262     Node prev = f;
263
264     /*: note InBucket: "v : prev..conts" from InContents, ContentsDefInv,
265         thisNotNull, thisType, Init, HCPProps;
266         note PrevNotNull: "prev ~= null" from InBucket, ConDef, ConNull; */
267
268     /*: "prev..conts" := "prev..conts - {v}"; */
269     /*: "contents" := "old contents - {v}" */
270
271     Node curr = prev.next;
272
273     /*: note PrevHidden: "prev : hidden" from NodeHidden1, thisNotNull, thisType,
274         PrevNotNull, Init, HCPProps; */
275
276     /*: note ConPreLoop: "ALL n. n : Node & n : alloc & n ~= null & n ~= prev -->
277         n..conts = {n..value} Un n..next..conts & (n..value ~: n..next..conts)" from
278         ConDef, FirstInjInv, Init, HCPProps, thisNotNull, thisType, PrevNotNull;
279         note ConUnchangedPreLoop: "ALL hs i. hs ~= this & hs : HashSet & hs : alloc
280             & hs..init & 0 <= i & i < hs..table..length --> hs..table.[i]..conts =
281             old (hs..table.[i]..conts)" from ElementInjInv, thisType, PrevNotNull,
282             Init, HCPProps; */
283     while /*: invariant "prev : Node & prev : alloc & prev ~= null & prev : hidden
284         & comment ''PrevCon'' (prev..conts = fieldRead (old conts) prev - {v}) &
285         comment ''PrevNot'' (prev..value ~: prev..next..conts) & comment
286         ''CurrProps'' (curr : Node & curr : alloc) & comment ''CurrNotNull'' (curr
287             ~= null) & comment ''PrevCurr'' (prev..next = curr & prev ~= curr) &
288         contents = old contents - {v} & v : curr..conts & comment ''ConDefInv'' (ALL
289             n. n : Node & n : alloc & n ~= null & n ~= prev --> n..conts = {n..value} Un
290             n..next..conts & n..value ~: n..next..conts) & comment ''ConLoop'' (ALL n.
291             n..conts = old (n..conts) | n..conts = old (n..conts) - {v}) & (null..conts
292             = {}) & comment ''FConInv'' (f..conts = (fieldRead (old conts) f) - {v}) &
293             comment ''ConUnchanged'' (ALL hs i. hs ~= this & hs : HashSet & hs : alloc &
294             hs..init & 0 <= i & i < hs..table..length --> hs..table.[i]..conts = old
295             (hs..table.[i]..conts))" */ (curr.value != v) {
296         /*: "curr..conts" := "curr..conts - {v}"; */
297
298         /*: note CurrCon: "curr..conts = fieldRead (old conts) curr - {v}";
299             note PrevIsNot: "prev..value ~= v";
300             note OldConDef: "fieldRead (old conts) prev = {prev..value} Un
301                 fieldRead (old conts) (prev..next)";
302             note PrevConDef: "prev..conts = {prev..value} Un prev..next..conts"
303                 from PrevCurr, PrevCon, CurrCon, OldConDef, PrevIsNot; */
304
305         prev = curr;
306         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 &
330         x..init";
331     note OldXContents: "fieldRead (old HashSet.contents) x = {w. w :
332         fieldRead (old conts) (arrayRead (old arrayState) (x..table) (h w
333         (x..table..length)))}" from ContentsDefHyp, ContentsDefInv; */
334     {
335         /*: assuming XNotThisHyp: "x ~= this";
336         note NotCurr: "ALL i. 0 <= i & i < x..table..length -->
337             x..table.[i] ~= curr";
338         note ConXUnchanged: "ALL i. 0 <= i & i < x..table..length -->
339             x..table.[i]..conts = fieldRead (old conts) (arrayRead (old
340             arrayState) (x..table) i)" from ContentsDefHyp, XNotThisHyp,
341             NotCurr, ConUnchanged;
342         note LengthLemma: "ALL w. 0 <= (h w (x..table..length)) & (h w
343             (x..table..length)) < (x..table..length)" from ContentsDefHyp,
344             HashInv;
345         note XNotThisCase: "x..contents = {w. w : x..table.[(h w
346             (x..table..length))].conts}" from XNotThisHyp, OldXContents,
347             LengthLemma, ConXUnchanged; */
348     }
349     {
350         /*: assuming XIsThisHyp: "x = this";
351         note OldContents: "old contents = {w. w : old (table.[(h w
352             (table..length))].conts)}"; */
353     {
354         /*: pickAny w::obj suchThat ForwHyp: "w : contents";
355         note NotCurr: "table.[(h w (table..length))] ~= curr" from
356             FirstInjInv, ContentsDefHyp, XIsThisHyp, PrevCurr,
357             CurrNotNull, HashInv;
358         note ForwCase: "w : table.[(h w (table..length))].conts" from
359             ForwHyp, OldContents, NotCurr, ConLoop forSuch w; */
360     }
361     {
362         /*: pickAny w::obj suchThat BackHyp: "w : table.[(h w
363             (table..length))].conts";
364         note NotCurr: "table.[(h w (table..length))] ~= curr" from
365             FirstInjInv, ContentsDefHyp, XIsThisHyp, PrevCurr,
366             CurrNotNull, HashInv;
367         note BackCase: "w : contents" from BackHyp, OldContents,
368             NotCurr, ConLoop, FConInv, HCProps forSuch w; */
369     }
370     /*: note XIsThisCase: "contents = {w. w : table.[(h w
371             (table..length))].conts}" from ForwCase, BackCase; */
372 }
373 /*: note ContentsDefPost: "x..contents = {w. w : x..table.[(h w
374             (x..table..length))].conts}" from XNotThisCase, XIsThisCase forSuch x;
375 */
376 }
377
378 /*: note ContentsPost: "contents = old contents - {v}" */
379 {
380     /*: pickAny x::obj */
381     {
382         /*: assuming CardHyp: "x : alloc & x : HashSet & x..init" */
383         {
384             /*: note ThisProps: "this : old alloc & this : HashSet & old init"
385                 */
386             /*: note OldCard: "old _size = card (old contents)" from ThisProps,
387                 CardInv */
388             /*: note NewLength: "_size = old _size - 1" */
389             /*: note NewNotInOld: "v : old contents" */
390             /*: note XIsThisCard: "_size = card (contents)" from OldCard,
391                 NewLength, NewNotInOld, ContentsPost */
392         }
393     }
394 }

```

```

369     /*: assuming XNotThisHyp: "x ~= this" */
370     /*: note XInOldAlloc: "x : old alloc" */
371     /*: note XOldInit: "x..(old HashSet.init)" */
372     /*: note OldCard: "x..(old HashSet._size) = card (x..(old
    HashSet.contents))" from CardHyp, XNotThisHyp, XInOldAlloc,
    XOldInit, CardInv */
373     /*: note XLengthEq: "x.._size = x..(old HashSet._size)" */
374     {
375         /*: localize */
376         /*: note XContentsForw: "ALL y. y : x..contents --> y : x..(old
    HashSet.contents)" */
377         /*: note XContentsBack: "ALL y. y : x..(old HashSet.contents)
    --> y : x..contents" */
378         /*: note XContentsUnchanged: "x..contents = x..(old
    HashSet.contents)" from XContentsForw, XContentsBack */
379     }
380     /*: note XNotThisCard: "x.._size = card (x..contents)" from
    XLengthEq, OldCard, XContentsUnchanged */
381 }
382 /*: note CardConc: "x.._size = card (x..contents)" from XIsThisCard,
    XNotThisCard */
383 }
384 /*: note CardPostCond: "x : alloc & x : HashSet & x..init --> x.._size =
    card (x..contents)" forSuch x */
385 }
386 }
387
388 private void _remove(Object v)
389 /*: 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" */
390 {
391     int hc = compute_hash(v);
392     Node f = table[hc];
393
394     /*: 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; */
395
396     if (f.value == v) {
397         removeFirst(v, hc);
398     } else {
399         removeFromBucket(v, hc);
400     }
401 }
402
403 private void _add(Object v)
404 /*: 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" */
405 {
406     int hc = compute_hash(v);
407     Node n = new /*: hidden */ Node();
408     n.value = v;
409     Node first = table[hc];
410     n.next = first;
411     /*: "n..conts" := "{v} Un first..conts"; */
412     table[hc] = n;
413     /*: "contents" := "(old contents) Un {v}"; */
414 }
415
416
417
418
419
420

```

```

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     {

```

```

466   /*: assuming h1: "hs : alloc & hs : HashSet & hs..init"; */
467   {
468     /*: pickAny i::int, w::obj;
469       note g1: "arrayRead (old arrayState) (hs..table) i ~= n";
470       note g2: "hs : old alloc"; */
471     {
472       /*: assuming h2: "0 <= i & i < hs..table..length & w :
473         hs..table.[i]..conts"; */
474       {
475         /*: assuming h3: "hs = this";
476           note c5: "h w (hs..table..length) = i" from c3, h1, h2,
477             h3, Coherence, g1, g2, HProp; */
478       }
479       {
480         /*: assuming h4: "hs ~= this";
481           note g3: "hs..table ~= table";
482           note c4: "h w (hs..table..length) = i" from c4, h1, h2,
483             h4, Coherence, g1, g2, g3; */
484       }
485       /*: note c3: "h w (hs..table..length) = i" from c4, c5; */
486     }
487     /*: note c2: "0 <= i & i < hs..table..length --> w :
488       hs..table.[i]..conts --> h w (hs..table..length) = i" forSuch i,
489       w; */
490   }
491   /*: note CohPost: "hs : alloc & hs : HashSet & hs..init --> (ALL i w. 0 <=
492     i & i < hs..table..length --> w : hs..table.[i]..conts --> h w
493     (hs..table..length) = i)" forSuch hs; */
494 }
495 {
496   /*: pickAny hs::obj; */
497   {
498     /*: assuming NewContentsHyp: "hs : alloc & hs : HashSet & hs..init";
499       note ThisContents: "contents = {w. w : table.[(h w
500         (table..length))].conts}" from ThisContents, HProp,
501         NewNotRefThisArr, HashInv, ContentsDefInv, ThisProps, Init; */
502     {
503       /*: assuming OtherHyp: "hs ~= this";
504         note OtherContents: "hs..contents = {w. w : hs..table.[(h w
505           (hs..table..length))].conts}" from OtherHyp,
506           NewContentsHyp, ContentsDefInvHash, NewNotHT,
507           TableInjInvHash, NewNotRefArray, TableNotNullHash, HashInv;
508         */
509     }
510     /*: cases "hs = this", "hs ~= this" for NewContentsCases: "hs..contents
511       = {w. w : hs..table.[(h w (hs..table..length))].conts}" from
512       ThisContents, OtherContents;
513       note "hs..contents = {w. w : hs..table.[(h w
514         (hs..table..length))].conts}"; */
515     }
516     /*: note NewContentsDef: "hs : alloc & hs : HashSet & hs..init -->
517       hs..contents = {w. w : hs..table.[(h w (hs..table..length))].conts}"
518       forSuch hs; */
519   }
520 }
521 /*: note OldNotRefByNext: "ALL x. first ~= null --> old (x..next) ~= first"
522 from FirstInjInv, Init, HCBounds, OldNotRefByNext, ThisProps;
523
524 note NewFirstInj: "theinv FirstInjInv" from FirstInjInv, NewNotRefByNext,
525 OldNotRefByNext, TableInjInv, OldNotRefInTable, HCBounds, NewFirstInj,
526 ThisProps, ElementInjInv, NewNotHT;

```



```

509     note NewNextInj: "theinv NextInjInv" from NextInjInv, OldNotRefByNext,
510         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                 */
520                 /*: note OldCard: "old _size = card (old contents)" from ThisProps,
521                 CardInv */
522                 /*: note NewLength: "_size = old _size + 1" */
523                 /*: note NewNotInOld: "v ~: old contents" */
524                 /*: note XIsThisCard: "_size = card (contents)" from OldCard,
525                 NewLength, NewNotInOld, ContentsPost */
526             }
527             {
528                 /*: assuming XNotThisHyp: "x ~= this" */
529                 /*: note XInOldAlloc: "x : old alloc" */
530                 /*: note XOldInit: "x..(old HashSet.init)" */
531                 /*: note OldCard: "x..(old HashSet._size) = card (x..(old
532                 HashSet.contents))" from CardHyp, XNotThisHyp, XInOldAlloc,
533                 XOldInit, CardInv */
534                 /*: note XLengthEq: "x.._size = x..(old HashSet._size)" */
535                 {
536                     /*: localize */
537                     /*: note XContentsForw: "ALL y. y : x..contents --> y : x..(old
538                     HashSet.contents)" */
539                     /*: note XContentsBack: "ALL y. y : x..(old HashSet.contents)
540                     --> y : x..contents" */
541                     /*: note XContentsUnchanged: "x..contents = x..(old
542                     HashSet.contents)" from XContentsForw, XContentsBack */
543                 }
544                 /*: note XNotThisCard: "x.._size = card (x..contents)" from
545                 XLengthEq, OldCard, XContentsUnchanged */
546             }
547             /*: note CardConc: "x.._size = card (x..contents)" from XIsThisCard,
548             XNotThisCard */
549         }
550         /*: note CardPostCond: "x : alloc & x : HashSet & x..init --> x.._size =
551         card (x..contents)" forSuch x */
552     }
553 }
554
555 public boolean add(Object v)
556 /*: requires "init & v ~= null"
557 modifies "contents", "size"
558 ensures "(v ~: old contents --> contents = old contents Un {v} & size = old
559 size + 1 & result) & (v : old contents --> contents = old contents & size =
560 old size & ~result)" */
561 {
562     if (_contains(v)) {
563         return false;
564     } else {
565         _add(v);
566         return true;
567     }
568 }
569
570 public int size()
571 /*: requires "init"
572 ensures "result = size" */

```

```

560     {
561         return _size;
562     }
563 }

```

### B.3.2 Commutativity Testing Methods

Listing 8. HashSetComm.java

```

1  class HashSetComm {
2      static void add_add_pre_s_0(HashSet sa, HashSet sb, Object v1, Object v2)
3      /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
4         & v2 ~= null &
5             sa..contents = sb..contents & sa..size = sb..size"
6         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
7         ensures "True" */
8      {
9         /*: assume "v1 ~= v2 | v1 : sa..contents" */
10         boolean r1a = sa.add(v1);
11         boolean r2a = sa.add(v2);
12
13         boolean r2b = sb.add(v2);
14         boolean r1b = sb.add(v1);
15
16         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
17             sb..size" */
18     }
19
20     static void add_add_pre_c_0(HashSet sa, HashSet sb, Object v1, Object v2)
21     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
22         & v2 ~= null &
23             sa..contents = sb..contents & sa..size = sb..size"
24         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
25         ensures "True" */
26     {
27         /*: assume "~(v1 ~= v2 | v1 : sa..contents)" */
28         boolean r1a = sa.add(v1);
29         boolean r2a = sa.add(v2);
30
31         boolean r2b = sb.add(v2);
32         boolean r1b = sb.add(v1);
33
34         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
35             sb..size)" */
36     }
37
38     static void add_add_between_s_1(HashSet sa, HashSet sb, Object v1, Object v2)
39     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
40         & v2 ~= null &
41             sa..contents = sb..contents & sa..size = sb..size"
42         modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
43         ensures "True" */
44     {
45         boolean r1a = sa.add(v1);
46         /*: assume "v1 ~= v2 | ~r1a" */
47         boolean r2a = sa.add(v2);
48
49         boolean r2b = sb.add(v2);
50         boolean r1b = sb.add(v1);
51
52         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
53             sb..size" */
54     }
55
56     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
116    & v2 ~= null &
117     sa..contents = sb..contents & sa..size = sb..size"
118    modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
119    ensures "True" */
120 {
121     /*: assume "~(v1 ~= v2 | v1 : sa..contents)" */
122     boolean r1a = sa.add(v1);
123     sa.add(v2);
124
125     sb.add(v2);
126     boolean r1b = sb.add(v1);
127
128     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
129 }
130
131 static void add_add_between_s_4(HashSet sa, HashSet sb, Object v1, Object v2)
132 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
133    & v2 ~= null &
134     sa..contents = sb..contents & sa..size = sb..size"
135    modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
136    ensures "True" */
137 {
138     boolean r1a = sa.add(v1);
139     /*: assume "v1 ~= v2 | ~r1a" */
140     sa.add(v2);
141
142     sb.add(v2);
143     boolean r1b = sb.add(v1);
144
145     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
146 }
147
148 static void add_add_between_c_4(HashSet sa, HashSet sb, Object v1, Object v2)
149 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
150    & v2 ~= null &
151     sa..contents = sb..contents & sa..size = sb..size"
152    modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
153    ensures "True" */
154 {
155     boolean r1a = sa.add(v1);
156     /*: assume "~(v1 ~= v2 | ~r1a)" */
157     sa.add(v2);
158
159     sb.add(v2);
160     boolean r1b = sb.add(v1);
161
162     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
163 }
164
165 static void add_add_post_s_5(HashSet sa, HashSet sb, Object v1, Object v2)
166 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
167    & v2 ~= null &
168     sa..contents = sb..contents & sa..size = sb..size"
169    modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
170    ensures "True" */
171 {
172     boolean r1a = sa.add(v1);
173     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
180    & v2 ~= null &
181       sa..contents = sb..contents & sa..size = sb..size"
182    modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
183    ensures "True" */
184 {
185     boolean r1a = sa.add(v1);
186     sa.add(v2);
187     /*: assume "~(v1 ~= v2 | ~r1a)" */
188
189     sb.add(v2);
190     boolean r1b = sb.add(v1);
191
192     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
193 }
194
195 static void add_contains_pre_s_6(HashSet sa, HashSet sb, Object v1, Object v2)
196 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
197    & v2 ~= null &
198       sa..contents = sb..contents & sa..size = sb..size"
199    modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
200    ensures "True" */
201 {
202     /*: assume "v1 ~= v2 | v1 : sa..contents" */
203     boolean r1a = sa.add(v1);
204     boolean r2a = sa.contains(v2);
205
206     boolean r2b = sb.contains(v2);
207     boolean r1b = sb.add(v1);
208
209     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
210        sb..size" */
211 }
212
213 static void add_contains_pre_c_6(HashSet sa, HashSet sb, Object v1, Object v2)
214 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
215    & 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     /*: assume "~(v1 ~= v2 | v1 : sa..contents)" */
221     boolean r1a = sa.add(v1);
222     boolean r2a = sa.contains(v2);
223
224     boolean r2b = sb.contains(v2);
225     boolean r1b = sb.add(v1);
226
227     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
228        sb..size)" */
229 }
230
231 static void add_contains_between_s_7(HashSet sa, HashSet sb, Object v1, Object v2)
232 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
233    & v2 ~= null &
234       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 =
240         sb..size" */
241 }
242
243 static void add_contains_between_c_7(HashSet sa, HashSet sb, Object v1, Object v2)
244 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
245     & v2 ~= null &
246         sa..contents = sb..contents & sa..size = sb..size"
247     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
248     ensures "True" */
249 {
250     boolean r1a = sa.add(v1);
251     /*: assume "~(v1 ~= v2 | ~r1a)" */
252     boolean r2a = sa.contains(v2);
253
254     boolean r2b = sb.contains(v2);
255     boolean r1b = sb.add(v1);
256
257     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
258         sb..size)" */
259 }
260
261 static void add_contains_post_s_8(HashSet sa, HashSet sb, Object v1, Object v2)
262 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
263     & 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.add(v1);
269     boolean r2a = sa.contains(v2);
270     /*: assume "v1 ~= v2 | ~r1a" */
271
272     boolean r2b = sb.contains(v2);
273     boolean r1b = sb.add(v1);
274
275     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
276         sb..size" */
277 }
278
279 static void add_contains_post_c_8(HashSet sa, HashSet sb, Object v1, Object v2)
280 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
281     & v2 ~= null &
282         sa..contents = sb..contents & sa..size = sb..size"
283     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
284     ensures "True" */
285 {
286     boolean r1a = sa.add(v1);
287     boolean r2a = sa.contains(v2);
288     /*: assume "~(v1 ~= v2 | ~r1a)" */
289
290     boolean r2b = sb.contains(v2);
291     boolean r1b = sb.add(v1);

```

```

287     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
288         sb..size)" */
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 =
352         sb..size)" */
353 }
354
355 static void add_remove_post_s_11(HashSet sa, HashSet sb, Object v1, Object v2)
356 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
357     & v2 ~= null &
358     sa..contents = sb..contents & sa..size = sb..size"
359     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
360     ensures "True" */
361 {
362     boolean r1a = sa.add(v1);
363     boolean r2a = sa.remove(v2);
364     /*: assume "v1 ~= v2" */
365
366     boolean r2b = sb.remove(v2);
367     boolean r1b = sb.add(v1);
368
369     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
370         sb..size" */
371 }
372
373 static void add_remove_post_c_11(HashSet sa, HashSet sb, Object v1, Object v2)
374 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
375     & v2 ~= 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     boolean r1a = sa.add(v1);
381     boolean r2a = sa.remove(v2);
382     /*: assume "~(v1 ~= v2)" */
383
384     boolean r2b = sb.remove(v2);
385     boolean r1b = sb.add(v1);
386
387     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
388         sb..size)" */
389 }
390
391 static void add_remove_pre_s_12(HashSet sa, HashSet sb, Object v1, Object v2)
392 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
393     & v2 ~= null &
394     sa..contents = sb..contents & sa..size = sb..size"
395     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
396     ensures "True" */
397 {
398     /*: assume "v1 ~= v2" */
399     boolean r1a = sa.add(v1);
400     sa.remove(v2);
401
402     sb.remove(v2);
403     boolean r1b = sb.add(v1);
404
405     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
406 }
407
408 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     int r2b = sb.size();
524     boolean r1b = sb.add(v1);
525
526     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
527         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     &
533         sa..contents = sb..contents & sa..size = sb..size"
534     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
535     ensures "True" */
536 {
537     boolean r1a = sa.add(v1);
538     /*: assume "~(r1a)" */
539     int r2a = sa.size();
540
541     int r2b = sb.size();
542     boolean r1b = sb.add(v1);
543
544     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
545         sb..size)" */
546 }
547
548 static void add_size_post_s_17(HashSet sa, HashSet sb, Object v1)
549 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
550     &
551         sa..contents = sb..contents & sa..size = sb..size"
552     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
553     ensures "True" */
554 {
555     boolean r1a = sa.add(v1);
556     int r2a = sa.size();
557     /*: assume "~r1a" */
558
559     int r2b = sb.size();
560     boolean r1b = sb.add(v1);
561
562     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
563         sb..size" */
564 }
565
566 static void add_size_post_c_17(HashSet sa, HashSet sb, Object v1)
567 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
568     &
569         sa..contents = sb..contents & sa..size = sb..size"
570     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
571     ensures "True" */
572 {
573     boolean r1a = sa.add(v1);
574     int r2a = sa.size();
575     /*: assume "~(r1a)" */
576
577     int r2b = sb.size();
578     boolean r1b = sb.add(v1);
579
580     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
581         sb..size)" */
582 }
583
584 static void add_add_pre_s_18(HashSet sa, HashSet sb, Object v1, Object v2)
585 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
586     & 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(HashSet sa, HashSet sb, Object v1, Object v2)
595 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
596     & v2 ~= null &
597         sa..contents = sb..contents & sa..size = sb..size"
598     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
599     ensures "True" */
600 {
601     /*: assume "~(v1 ~= v2 | v1 : sa..contents)" */
602     sa.add(v1);
603     boolean r2a = sa.add(v2);
604
605     boolean r2b = sb.add(v2);
606     sb.add(v1);
607
608     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
609 }
610
611 static void add_add_between_s_19(HashSet sa, HashSet sb, Object v1, Object v2)
612 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
613     & v2 ~= null &
614         sa..contents = sb..contents & sa..size = sb..size"
615     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
616     ensures "True" */
617 {
618     sa.add(v1);
619     /*: assume "v1 ~= v2 | v1 : sa..(old contents)" */
620     boolean r2a = sa.add(v2);
621
622     boolean r2b = sb.add(v2);
623     sb.add(v1);
624
625     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
626 }
627
628 static void add_add_between_c_19(HashSet sa, HashSet sb, Object v1, Object v2)
629 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
630     & v2 ~= null &
631         sa..contents = sb..contents & sa..size = sb..size"
632     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
633     ensures "True" */
634 {
635     sa.add(v1);
636     /*: assume "~(v1 ~= v2 | v1 : sa..(old contents))" */
637     boolean r2a = sa.add(v2);
638
639     boolean r2b = sb.add(v2);
640     sb.add(v1);
641
642     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
643 }

```

```

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

```

```

703     /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
704 }
705
706 static void add_add_between_s_22(HashSet sa, HashSet sb, Object v1, Object v2)
707 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashSet sa, HashSet sb, Object v1, Object v2)
723 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashSet sa, HashSet sb, Object v1, Object v2)
739 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashSet sa, HashSet sb, Object v1, Object v2)
755 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashSet sa, HashSet sb, Object v1, Object v2)
771 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
772    & v2 ~= null &
773       sa..contents = sb..contents & sa..size = sb..size"
774    modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
775    ensures "True" */
776 {
777     /*: assume "v1 ~= v2 | v1 : sa..contents" */
778     sa.add(v1);
779     boolean r2a = sa.contains(v2);
780
781     boolean r2b = sb.contains(v2);
782     sb.add(v1);
783
784     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
785 }
786
787 static void add_contains_pre_c_24(HashSet sa, HashSet sb, Object v1, Object v2)
788 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
789    & v2 ~= null &
790       sa..contents = sb..contents & sa..size = sb..size"
791    modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
792    ensures "True" */
793 {
794     /*: assume "~(v1 ~= v2 | v1 : sa..contents)" */
795     sa.add(v1);
796     boolean r2a = sa.contains(v2);
797
798     boolean r2b = sb.contains(v2);
799     sb.add(v1);
800
801     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
802 }
803
804 static void add_contains_between_s_25(HashSet sa, HashSet sb, Object v1, Object v2)
805 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
806    & v2 ~= null &
807       sa..contents = sb..contents & sa..size = sb..size"
808    modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
809    ensures "True" */
810 {
811     sa.add(v1);
812     /*: assume "v1 ~= v2 | v1 : sa..(old contents)" */
813     boolean r2a = sa.contains(v2);
814
815     boolean r2b = sb.contains(v2);
816     sb.add(v1);
817
818     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
819 }
820
821 static void add_contains_between_c_25(HashSet sa, HashSet sb, Object v1, Object v2)
822 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
823    & v2 ~= null &
824       sa..contents = sb..contents & sa..size = sb..size"
825    modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
826    ensures "True" */
827 {
828     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 &
    sa..contents = sb..contents & sa..size = sb..size"
    modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
    ensures "True" */
900 {
901     sa.add(v1);
902     /*: assume "v1 ~= v2" */
903     boolean r2a = sa.remove(v2);
904
905     boolean r2b = sb.remove(v2);
906     sb.add(v1);
907
908     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
909 }
910
911 static void add_remove_between_c_28(HashSet sa, HashSet sb, Object v1, Object v2)
912 /*: 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" */
913 {
914     sa.add(v1);
915     /*: assume "~(v1 ~= v2)" */
916     boolean r2a = sa.remove(v2);
917
918     boolean r2b = sb.remove(v2);
919     sb.add(v1);
920
921     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
922 }
923
924 static void add_remove_post_s_29(HashSet sa, HashSet sb, Object v1, Object v2)
925 /*: 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" */
926 {
927     sa.add(v1);
928     boolean r2a = sa.remove(v2);
929     /*: assume "v1 ~= v2" */
930
931     boolean r2b = sb.remove(v2);
932     sb.add(v1);
933
934     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
935 }
936
937 static void add_remove_post_c_29(HashSet sa, HashSet sb, Object v1, Object v2)

```

```

947  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
    & v2 ~= null &
948      sa..contents = sb..contents & sa..size = sb..size"
949  modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
950  ensures "True" */
951  {
952      sa.add(v1);
953      boolean r2a = sa.remove(v2);
954      /*: assume "~(v1 ~= v2)" */
955
956      boolean r2b = sb.remove(v2);
957      sb.add(v1);
958
959      /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
960  }
961
962  static void add_remove_pre_s_30(HashSet sa, HashSet sb, Object v1, Object v2)
963  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
    & v2 ~= null &
964      sa..contents = sb..contents & sa..size = sb..size"
965  modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
966  ensures "True" */
967  {
968      /*: assume "v1 ~= v2" */
969      sa.add(v1);
970      sa.remove(v2);
971
972      sb.remove(v2);
973      sb.add(v1);
974
975      /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
976  }
977
978  static void add_remove_pre_c_30(HashSet sa, HashSet sb, Object v1, Object v2)
979  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
    & v2 ~= null &
980      sa..contents = sb..contents & sa..size = sb..size"
981  modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
982  ensures "True" */
983  {
984      /*: assume "~(v1 ~= v2)" */
985      sa.add(v1);
986      sa.remove(v2);
987
988      sb.remove(v2);
989      sb.add(v1);
990
991      /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
992  }
993
994  static void add_remove_between_s_31(HashSet sa, HashSet sb, Object v1, Object v2)
995  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
    & v2 ~= null &
996      sa..contents = sb..contents & sa..size = sb..size"
997  modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
998  ensures "True" */
999  {
1000     sa.add(v1);
1001     /*: assume "v1 ~= v2" */
1002     sa.remove(v2);
1003
1004     sb.remove(v2);
1005     sb.add(v1);
1006
1007     /*: 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();

```

```

1069     sb.add(v1);
1070
1071     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
1072 }
1073
1074 static void add_size_pre_c_33(HashSet sa, HashSet sb, Object v1)
1075 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
    &
1076         sa..contents = sb..contents & sa..size = sb..size"
1077     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1078     ensures "True" */
1079 {
1080     /*: assume "~(v1 : sa..contents)" */
1081     sa.add(v1);
1082     int r2a = sa.size();
1083
1084     int r2b = sb.size();
1085     sb.add(v1);
1086
1087     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
1088 }
1089
1090 static void add_size_between_s_34(HashSet sa, HashSet sb, Object v1)
1091 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
    &
1092         sa..contents = sb..contents & sa..size = sb..size"
1093     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1094     ensures "True" */
1095 {
1096     sa.add(v1);
1097     /*: assume "v1 : sa..(old contents)" */
1098     int r2a = sa.size();
1099
1100     int r2b = sb.size();
1101     sb.add(v1);
1102
1103     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
1104 }
1105
1106 static void add_size_between_c_34(HashSet sa, HashSet sb, Object v1)
1107 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
    &
1108         sa..contents = sb..contents & sa..size = sb..size"
1109     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1110     ensures "True" */
1111 {
1112     sa.add(v1);
1113     /*: assume "~(v1 : sa..(old contents))" */
1114     int r2a = sa.size();
1115
1116     int r2b = sb.size();
1117     sb.add(v1);
1118
1119     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
1120 }
1121
1122 static void add_size_post_s_35(HashSet sa, HashSet sb, Object v1)
1123 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
    &
1124         sa..contents = sb..contents & sa..size = sb..size"
1125     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1126     ensures "True" */
1127 {
1128     sa.add(v1);
1129     int r2a = sa.size();

```

```

1130     /*: assume "v1 : sa..(old contents)" */
1131
1132     int r2b = sb.size();
1133     sb.add(v1);
1134
1135     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
1136 }
1137
1138 static void add_size_post_c_35(HashSet sa, HashSet sb, Object v1)
1139 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1140 &
1141     sa..contents = sb..contents & sa..size = sb..size"
1142 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1143 ensures "True" */
1144 {
1145     sa.add(v1);
1146     int r2a = sa.size();
1147     /*: assume "~(v1 : sa..(old contents))" */
1148
1149     int r2b = sb.size();
1150     sb.add(v1);
1151
1152     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
1153 }
1154
1155 static void contains_add_pre_s_36(HashSet sa, HashSet sb, Object v1, Object v2)
1156 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1157 & v2 ~= 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     /*: assume "v1 ~= v2 | v1 : sa..contents" */
1163     boolean r1a = sa.contains(v1);
1164     boolean r2a = sa.add(v2);
1165
1166     boolean r2b = sb.add(v2);
1167     boolean r1b = sb.contains(v1);
1168
1169     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1170 sb..size" */
1171 }
1172
1173 static void contains_add_pre_c_36(HashSet sa, HashSet sb, Object v1, Object v2)
1174 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1175 & v2 ~= null &
1176     sa..contents = sb..contents & sa..size = sb..size"
1177 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1178 ensures "True" */
1179 {
1180     /*: assume "~(v1 ~= v2 | v1 : sa..contents)" */
1181     boolean r1a = sa.contains(v1);
1182     boolean r2a = sa.add(v2);
1183
1184     boolean r2b = sb.add(v2);
1185     boolean r1b = sb.contains(v1);
1186
1187     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1188 sb..size)" */
1189 }
1190
1191 static void contains_add_between_s_37(HashSet sa, HashSet sb, Object v1, Object v2)
1192 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1193 & v2 ~= null &
1194     sa..contents = sb..contents & sa..size = sb..size"

```

```

1189     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1190     ensures "True" */
1191 {
1192     boolean r1a = sa.contains(v1);
1193     /*: assume "v1 ~= v2 | r1a" */
1194     boolean r2a = sa.add(v2);
1195
1196     boolean r2b = sb.add(v2);
1197     boolean r1b = sb.contains(v1);
1198
1199     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1200         sb..size" */
1201 }
1202
1203 static void contains_add_between_c_37(HashSet sa, HashSet sb, Object v1, Object v2)
1204 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1205     & v2 ~= 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     boolean r1a = sa.contains(v1);
1211     /*: assume "~(v1 ~= v2 | r1a)" */
1212     boolean r2a = sa.add(v2);
1213
1214     boolean r2b = sb.add(v2);
1215     boolean r1b = sb.contains(v1);
1216
1217     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1218         sb..size)" */
1219 }
1220
1221 static void contains_add_post_s_38(HashSet sa, HashSet sb, Object v1, Object v2)
1222 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1223     & v2 ~= null &
1224         sa..contents = sb..contents & sa..size = sb..size"
1225     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1226     ensures "True" */
1227 {
1228     boolean r1a = sa.contains(v1);
1229     boolean r2a = sa.add(v2);
1230     /*: assume "v1 ~= v2 | r1a" */
1231
1232     boolean r2b = sb.add(v2);
1233     boolean r1b = sb.contains(v1);
1234
1235     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1236         sb..size" */
1237 }
1238
1239 static void contains_add_post_c_38(HashSet sa, HashSet sb, Object v1, Object v2)
1240 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1241     & v2 ~= null &
1242         sa..contents = sb..contents & sa..size = sb..size"
1243     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1244     ensures "True" */
1245 {
1246     boolean r1a = sa.contains(v1);
1247     boolean r2a = sa.add(v2);
1248     /*: assume "~(v1 ~= v2 | r1a)" */
1249
1250     boolean r2b = sb.add(v2);
1251     boolean r1b = sb.contains(v1);

```

```

1247     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1248         sb..size)" */
1249 }
1250 static void contains_add_pre_s_39(HashSet sa, HashSet sb, Object v1, Object v2)
1251 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1252     & v2 ~= null &
1253         sa..contents = sb..contents & sa..size = sb..size"
1254     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1255     ensures "True" */
1256 {
1257     /*: assume "v1 ~= v2 | v1 : sa..contents" */
1258     boolean r1a = sa.contains(v1);
1259     sa.add(v2);
1260
1261     sb.add(v2);
1262     boolean r1b = sb.contains(v1);
1263
1264     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1265 }
1266 static void contains_add_pre_c_39(HashSet sa, HashSet sb, Object v1, Object v2)
1267 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1268     & v2 ~= null &
1269         sa..contents = sb..contents & sa..size = sb..size"
1270     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1271     ensures "True" */
1272 {
1273     /*: assume "~(v1 ~= v2 | v1 : sa..contents)" */
1274     boolean r1a = sa.contains(v1);
1275     sa.add(v2);
1276
1277     sb.add(v2);
1278     boolean r1b = sb.contains(v1);
1279
1280     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1281 }
1282 static void contains_add_between_s_40(HashSet sa, HashSet sb, Object v1, Object v2)
1283 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1284     & v2 ~= null &
1285         sa..contents = sb..contents & sa..size = sb..size"
1286     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1287     ensures "True" */
1288 {
1289     boolean r1a = sa.contains(v1);
1290     /*: assume "v1 ~= v2 | r1a" */
1291     sa.add(v2);
1292
1293     sb.add(v2);
1294     boolean r1b = sb.contains(v1);
1295
1296     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1297 }
1298 static void contains_add_between_c_40(HashSet sa, HashSet sb, Object v1, Object v2)
1299 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1300     & v2 ~= null &
1301         sa..contents = sb..contents & sa..size = sb..size"
1302     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1303     ensures "True" */
1304 {
1305     boolean r1a = sa.contains(v1);
1306     /*: assume "~(v1 ~= v2 | r1a)" */
1307     sa.add(v2);

```

```

1307     sb.add(v2);
1308     boolean r1b = sb.contains(v1);
1309
1310     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1311 }
1312
1313
1314 static void contains_add_post_s_41(HashSet sa, HashSet sb, Object v1, Object v2)
1315 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
    & v2 ~= null &
1316         sa..contents = sb..contents & sa..size = sb..size"
1317     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1318     ensures "True" */
1319 {
1320     boolean r1a = sa.contains(v1);
1321     sa.add(v2);
1322     /*: assume "v1 ~= v2 | r1a" */
1323
1324     sb.add(v2);
1325     boolean r1b = sb.contains(v1);
1326
1327     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1328 }
1329
1330 static void contains_add_post_c_41(HashSet sa, HashSet sb, Object v1, Object v2)
1331 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
    & v2 ~= null &
1332         sa..contents = sb..contents & sa..size = sb..size"
1333     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1334     ensures "True" */
1335 {
1336     boolean r1a = sa.contains(v1);
1337     sa.add(v2);
1338     /*: assume "~(v1 ~= v2 | r1a)" */
1339
1340     sb.add(v2);
1341     boolean r1b = sb.contains(v1);
1342
1343     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1344 }
1345
1346 static void contains_contains_pre_s_42(HashSet sa, HashSet sb, Object v1, Object v2)
1347 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
    & v2 ~= null &
1348         sa..contents = sb..contents & sa..size = sb..size"
1349     ensures "True" */
1350 {
1351     /*: assume "True" */
1352     boolean r1a = sa.contains(v1);
1353     boolean r2a = sa.contains(v2);
1354
1355     boolean r2b = sb.contains(v2);
1356     boolean r1b = sb.contains(v1);
1357
1358     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
    sb..size" */
1359 }
1360
1361 static void contains_contains_pre_c_42(HashSet sa, HashSet sb, Object v1, Object v2)
1362 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
    & v2 ~= null &
1363         sa..contents = sb..contents & sa..size = sb..size"
1364     ensures "True" */
1365 {
1366     /*: 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

```

```

1421 static void contains_contains_post_c_44(HashSet sa, HashSet sb, Object v1, Object
1422 v2)
1423 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1424 & v2 ~= null &
1425         sa..contents = sb..contents & sa..size = sb..size"
1426 ensures "True" */
1427 {
1428     boolean r1a = sa.contains(v1);
1429     boolean r2a = sa.contains(v2);
1430     /*: assume "~(True)" */
1431
1432     boolean r2b = sb.contains(v2);
1433     boolean r1b = sb.contains(v1);
1434
1435     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1436         sb..size)" */
1437 }
1438
1439 static void contains_remove_pre_s_45(HashSet sa, HashSet sb, Object v1, Object v2)
1440 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1441 & v2 ~= null &
1442         sa..contents = sb..contents & sa..size = sb..size"
1443 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1444 ensures "True" */
1445 {
1446     /*: assume "v1 ~= v2 | v1 ~: sa..contents" */
1447     boolean r1a = sa.contains(v1);
1448     boolean r2a = sa.remove(v2);
1449
1450     boolean r2b = sb.remove(v2);
1451     boolean r1b = sb.contains(v1);
1452
1453     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1454         sb..size" */
1455 }
1456
1457 static void contains_remove_pre_c_45(HashSet sa, HashSet sb, Object v1, Object v2)
1458 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1459 & v2 ~= null &
1460         sa..contents = sb..contents & sa..size = sb..size"
1461 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1462 ensures "True" */
1463 {
1464     /*: assume "~(v1 ~= v2 | v1 ~: sa..contents)" */
1465     boolean r1a = sa.contains(v1);
1466     boolean r2a = sa.remove(v2);
1467
1468     boolean r2b = sb.remove(v2);
1469     boolean r1b = sb.contains(v1);
1470
1471     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1472         sb..size)" */
1473 }
1474
1475 static void contains_remove_between_s_46(HashSet sa, HashSet sb, Object v1, Object
1476 v2)
1477 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1478 & v2 ~= null &
1479         sa..contents = sb..contents & sa..size = sb..size"
1480 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1481 ensures "True" */
1482 {
1483     boolean r1a = sa.contains(v1);
1484     /*: assume "v1 ~= v2 | ~r1a" */
1485     boolean r2a = sa.remove(v2);

```

```

1477     boolean r2b = sb.remove(v2);
1478     boolean r1b = sb.contains(v1);
1479
1480     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1481         sb..size" */
1482 }
1483
1484 static void contains_remove_between_c_46(HashSet sa, HashSet sb, Object v1, Object
1485     v2)
1486 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1487     & v2 ~= 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     boolean r1a = sa.contains(v1);
1493     /*: assume "~(v1 ~= v2 | ~r1a)" */
1494     boolean r2a = sa.remove(v2);
1495
1496     boolean r2b = sb.remove(v2);
1497     boolean r1b = sb.contains(v1);
1498
1499     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1500         sb..size)" */
1501 }
1502
1503 static void contains_remove_post_s_47(HashSet sa, HashSet sb, Object v1, Object v2)
1504 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1505     & v2 ~= null &
1506         sa..contents = sb..contents & sa..size = sb..size"
1507     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1508     ensures "True" */
1509 {
1510     boolean r1a = sa.contains(v1);
1511     boolean r2a = sa.remove(v2);
1512     /*: assume "v1 ~= v2 | ~r1a" */
1513
1514     boolean r2b = sb.remove(v2);
1515     boolean r1b = sb.contains(v1);
1516
1517     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1518         sb..size" */
1519 }
1520
1521 static void contains_remove_post_c_47(HashSet sa, HashSet sb, Object v1, Object v2)
1522 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1523     & v2 ~= null &
1524         sa..contents = sb..contents & sa..size = sb..size"
1525     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1526     ensures "True" */
1527 {
1528     boolean r1a = sa.contains(v1);
1529     boolean r2a = sa.remove(v2);
1530     /*: assume "~(v1 ~= v2 | ~r1a)" */
1531
1532     boolean r2b = sb.remove(v2);
1533     boolean r1b = sb.contains(v1);
1534
1535     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1536         sb..size)" */
1537 }
1538
1539 static void contains_remove_pre_s_48(HashSet sa, HashSet sb, Object v1, Object v2)

```

```

1533  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
      & 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(HashSet sa, HashSet sb, Object v1, Object v2)
1549  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
      & 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(HashSet sa, HashSet sb, Object v1, Object
      v2)
1565  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
      & 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(HashSet sa, HashSet sb, Object v1, Object
      v2)
1581  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
      & 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
1590      sb.remove(v2);
1591      boolean r1b = sb.contains(v1);

```

```

1592     /*: assert "~(ria = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1593 }
1594
1595 static void contains_remove_post_s_50(HashSet sa, HashSet sb, Object v1, Object v2)
1596 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1597    & 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
1614    & v2 ~= null &
1615        sa..contents = sb..contents & sa..size = sb..size"
1616    modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1617    ensures "True" */
1618 {
1619     boolean r1a = sa.contains(v1);
1620     sa.remove(v2);
1621     /*: assume "~(v1 ~= v2 | ~r1a)" */
1622
1623     sb.remove(v2);
1624     boolean r1b = sb.contains(v1);
1625
1626     /*: assert "~(ria = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1627 }
1628
1629 static void contains_size_pre_s_51(HashSet sa, HashSet sb, Object v1)
1630 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1631    &
1632        sa..contents = sb..contents & sa..size = sb..size"
1633    ensures "True" */
1634 {
1635     /*: assume "True" */
1636     boolean r1a = sa.contains(v1);
1637     int r2a = sa.size();
1638
1639     int r2b = sb.size();
1640     boolean r1b = sb.contains(v1);
1641
1642     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1643        sb..size" */
1644 }
1645
1646 static void contains_size_pre_c_51(HashSet sa, HashSet sb, Object v1)
1647 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1648    &
1649        sa..contents = sb..contents & sa..size = sb..size"
1650    ensures "True" */
1651 {
1652     /*: assume "~(True)" */
1653     boolean r1a = sa.contains(v1);
1654     int r2a = sa.size();

```

```

1652     int r2b = sb.size();
1653     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(HashSet sa, HashSet sb, Object v1)
1659 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        &
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(HashSet sa, HashSet sb, Object v1)
1674 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        &
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(HashSet sa, HashSet sb, Object v1)
1689 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        &
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(HashSet sa, HashSet sb, Object v1)
1704 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
        &
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
1715     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
        sb..size)" */
1716 }
1717
1718 static void remove_add_pre_s_54(HashSet sa, HashSet sb, Object v1, Object v2)
1719 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
    & v2 ~= null &
1720         sa..contents = sb..contents & sa..size = sb..size"
1721     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1722     ensures "True" */
1723 {
1724     /*: assume "v1 ~= v2" */
1725     boolean r1a = sa.remove(v1);
1726     boolean r2a = sa.add(v2);
1727
1728     boolean r2b = sb.add(v2);
1729     boolean r1b = sb.remove(v1);
1730
1731     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
        sb..size" */
1732 }
1733
1734 static void remove_add_pre_c_54(HashSet sa, HashSet sb, Object v1, Object v2)
1735 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
    & v2 ~= null &
1736         sa..contents = sb..contents & sa..size = sb..size"
1737     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1738     ensures "True" */
1739 {
1740     /*: assume "~(v1 ~= v2)" */
1741     boolean r1a = sa.remove(v1);
1742     boolean r2a = sa.add(v2);
1743
1744     boolean r2b = sb.add(v2);
1745     boolean r1b = sb.remove(v1);
1746
1747     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
        sb..size)" */
1748 }
1749
1750 static void remove_add_between_s_55(HashSet sa, HashSet sb, Object v1, Object v2)
1751 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
    & v2 ~= null &
1752         sa..contents = sb..contents & sa..size = sb..size"
1753     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1754     ensures "True" */
1755 {
1756     boolean r1a = sa.remove(v1);
1757     /*: assume "v1 ~= v2" */
1758     boolean r2a = sa.add(v2);
1759
1760     boolean r2b = sb.add(v2);
1761     boolean r1b = sb.remove(v1);
1762
1763     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
        sb..size" */
1764 }
1765
1766 static void remove_add_between_c_55(HashSet sa, HashSet sb, Object v1, Object v2)

```

```

1767  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1768      & v2 ~= null &
1769          sa..contents = sb..contents & sa..size = sb..size"
1770  modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1771  ensures "True" */
1772  {
1773      boolean r1a = sa.remove(v1);
1774      /*: assume "~(v1 ~= v2)" */
1775      boolean r2a = sa.add(v2);
1776
1777      boolean r2b = sb.add(v2);
1778      boolean r1b = sb.remove(v1);
1779
1780      /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1781          sb..size)" */
1782  }
1783
1784  static void remove_add_post_s_56(HashSet sa, HashSet sb, Object v1, Object v2)
1785  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1786      & v2 ~= null &
1787          sa..contents = sb..contents & sa..size = sb..size"
1788  modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1789  ensures "True" */
1790  {
1791      boolean r1a = sa.remove(v1);
1792      boolean r2a = sa.add(v2);
1793      /*: assume "v1 ~= v2" */
1794
1795      boolean r2b = sb.add(v2);
1796      boolean r1b = sb.remove(v1);
1797
1798      /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1799          sb..size" */
1800  }
1801
1802  static void remove_add_post_c_56(HashSet sa, HashSet sb, Object v1, Object v2)
1803  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1804      & v2 ~= null &
1805          sa..contents = sb..contents & sa..size = sb..size"
1806  modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1807  ensures "True" */
1808  {
1809      boolean r1a = sa.remove(v1);
1810      boolean r2a = sa.add(v2);
1811      /*: assume "~(v1 ~= v2)" */
1812
1813      boolean r2b = sb.add(v2);
1814      boolean r1b = sb.remove(v1);
1815
1816      /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1817          sb..size)" */
1818  }
1819
1820  static void remove_add_pre_s_57(HashSet sa, HashSet sb, Object v1, Object v2)
1821  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1822      & v2 ~= null &
1823          sa..contents = sb..contents & sa..size = sb..size"
1824  modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1825  ensures "True" */
1826  {
1827      /*: assume "v1 ~= v2" */
1828      boolean r1a = sa.remove(v1);
1829      sa.add(v2);
1830
1831      sb.add(v2);

```



```

1825     boolean r1b = sb.remove(v1);
1826
1827     /*: assert "ria = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1828 }
1829
1830 static void remove_add_pre_c_57(HashSet sa, HashSet sb, Object v1, Object v2)
1831 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1832    & v2 ~= null &
1833       sa..contents = sb..contents & sa..size = sb..size"
1834    modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1835    ensures "True" */
1836 {
1837     /*: assume "~(v1 ~= v2)" */
1838     boolean r1a = sa.remove(v1);
1839     sa.add(v2);
1840
1841     sb.add(v2);
1842     boolean r1b = sb.remove(v1);
1843
1844     /*: assert "~(ria = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1845 }
1846
1847 static void remove_add_between_s_58(HashSet sa, HashSet sb, Object v1, Object v2)
1848 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1849    & v2 ~= null &
1850       sa..contents = sb..contents & sa..size = sb..size"
1851    modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1852    ensures "True" */
1853 {
1854     boolean r1a = sa.remove(v1);
1855     /*: assume "v1 ~= v2" */
1856     sa.add(v2);
1857
1858     sb.add(v2);
1859     boolean r1b = sb.remove(v1);
1860
1861     /*: assert "ria = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1862 }
1863
1864 static void remove_add_between_c_58(HashSet sa, HashSet sb, Object v1, Object v2)
1865 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1866    & v2 ~= null &
1867       sa..contents = sb..contents & sa..size = sb..size"
1868    modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1869    ensures "True" */
1870 {
1871     boolean r1a = sa.remove(v1);
1872     /*: assume "~(v1 ~= v2)" */
1873     sa.add(v2);
1874
1875     sb.add(v2);
1876     boolean r1b = sb.remove(v1);
1877
1878     /*: assert "~(ria = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1879 }
1880
1881 static void remove_add_post_s_59(HashSet sa, HashSet sb, Object v1, Object v2)
1882 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1883    & v2 ~= null &
1884       sa..contents = sb..contents & sa..size = sb..size"
1885    modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1886    ensures "True" */
1887 {
1888     boolean r1a = sa.remove(v1);
1889     sa.add(v2);

```

```

1886     /*: assume "v1 ~= v2" */
1887
1888     sb.add(v2);
1889     boolean r1b = sb.remove(v1);
1890
1891     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1892 }
1893
1894 static void remove_add_post_c_59(HashSet sa, HashSet sb, Object v1, Object v2)
1895 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1896     & v2 ~= null &
1897         sa..contents = sb..contents & sa..size = sb..size"
1898     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1899     ensures "True" */
1900 {
1901     boolean r1a = sa.remove(v1);
1902     sa.add(v2);
1903     /*: assume "~(v1 ~= v2)" */
1904
1905     sb.add(v2);
1906     boolean r1b = sb.remove(v1);
1907
1908     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1909 }
1910
1911 static void remove_contains_pre_s_60(HashSet sa, HashSet sb, Object v1, Object v2)
1912 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1913     & v2 ~= null &
1914         sa..contents = sb..contents & sa..size = sb..size"
1915     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1916     ensures "True" */
1917 {
1918     /*: assume "v1 ~= v2 | v1 ~: sa..contents" */
1919     boolean r1a = sa.remove(v1);
1920     boolean r2a = sa.contains(v2);
1921
1922     boolean r2b = sb.contains(v2);
1923     boolean r1b = sb.remove(v1);
1924
1925     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1926         sb..size" */
1927 }
1928
1929 static void remove_contains_pre_c_60(HashSet sa, HashSet sb, Object v1, Object v2)
1930 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1931     & v2 ~= null &
1932         sa..contents = sb..contents & sa..size = sb..size"
1933     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1934     ensures "True" */
1935 {
1936     /*: assume "~(v1 ~= v2 | v1 ~: sa..contents)" */
1937     boolean r1a = sa.remove(v1);
1938     boolean r2a = sa.contains(v2);
1939
1940     boolean r2b = sb.contains(v2);
1941     boolean r1b = sb.remove(v1);
1942
1943     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1944         sb..size)" */
1945 }
1946
1947 static void remove_contains_between_s_61(HashSet sa, HashSet sb, Object v1, Object
1948     v2)
1949 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
1950     & 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     /*: assert "~(ria = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2003         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 "ria = 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 "~(ria = 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 "ria = 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" */

```

```

2059 {
2060     boolean r1a = sa.remove(v1);
2061     /*: assume "(v1 ~= v2 | ~r1a)" */
2062     boolean r2a = sa.remove(v2);
2063
2064     boolean r2b = sb.remove(v2);
2065     boolean r1b = sb.remove(v1);
2066
2067     /*: assert "(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
        sb..size)" */
2068 }
2069
2070 static void remove_remove_post_s_65(HashSet sa, HashSet sb, Object v1, Object v2)
2071 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
    & v2 ~= null &
2072         sa..contents = sb..contents & sa..size = sb..size"
2073     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2074     ensures "True" */
2075 {
2076     boolean r1a = sa.remove(v1);
2077     boolean r2a = sa.remove(v2);
2078     /*: assume "(v1 ~= v2 | ~r1a)" */
2079
2080     boolean r2b = sb.remove(v2);
2081     boolean r1b = sb.remove(v1);
2082
2083     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
        sb..size" */
2084 }
2085
2086 static void remove_remove_post_c_65(HashSet sa, HashSet sb, Object v1, Object v2)
2087 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashSet sa, HashSet sb, Object v1, Object v2)
2103 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashSet sa, HashSet sb, Object v1, Object v2)
2119 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashSet sa, HashSet sb, Object v1, Object v2)
2135 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashSet sa, HashSet sb, Object v1, Object v2)
2151 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashSet sa, HashSet sb, Object v1, Object v2)
2167 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashSet sa, HashSet sb, Object v1, Object v2)
2183 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashSet sa, HashSet sb, Object v1)
2199 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashSet sa, HashSet sb, Object v1)
2215 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashSet sa, HashSet sb, Object v1)
2231 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashSet sa, HashSet sb, Object v1)
2247 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashSet sa, HashSet sb, Object v1)
2263 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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
2275     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
        sb..size" */
2276 }
2277
2278 static void remove_size_post_c_71(HashSet sa, HashSet sb, Object v1)
2279 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
    &
2280         sa..contents = sb..contents & sa..size = sb..size"
2281 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2282 ensures "True" */
2283 {
2284     boolean r1a = sa.remove(v1);
2285     int r2a = sa.size();
2286     /*: assume "~(r1a)" */
2287
2288     int r2b = sb.size();
2289     boolean r1b = sb.remove(v1);
2290
2291     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
        sb..size)" */
2292 }
2293
2294 static void remove_add_pre_s_72(HashSet sa, HashSet sb, Object v1, Object v2)

```



```

2295  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
      & v2 ~= null &
2296          sa..contents = sb..contents & sa..size = sb..size"
2297  modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2298  ensures "True" */
2299  {
2300    /*: assume "v1 ~= v2" */
2301    sa.remove(v1);
2302    boolean r2a = sa.add(v2);
2303
2304    boolean r2b = sb.add(v2);
2305    sb.remove(v1);
2306
2307    /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2308  }
2309
2310  static void remove_add_pre_c_72(HashSet sa, HashSet sb, Object v1, Object v2)
2311  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
      & v2 ~= null &
2312          sa..contents = sb..contents & sa..size = sb..size"
2313  modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2314  ensures "True" */
2315  {
2316    /*: assume "~(v1 ~= v2)" */
2317    sa.remove(v1);
2318    boolean r2a = sa.add(v2);
2319
2320    boolean r2b = sb.add(v2);
2321    sb.remove(v1);
2322
2323    /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2324  }
2325
2326  static void remove_add_between_s_73(HashSet sa, HashSet sb, Object v1, Object v2)
2327  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
      & v2 ~= null &
2328          sa..contents = sb..contents & sa..size = sb..size"
2329  modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2330  ensures "True" */
2331  {
2332    sa.remove(v1);
2333    /*: assume "v1 ~= v2" */
2334    boolean r2a = sa.add(v2);
2335
2336    boolean r2b = sb.add(v2);
2337    sb.remove(v1);
2338
2339    /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2340  }
2341
2342  static void remove_add_between_c_73(HashSet sa, HashSet sb, Object v1, Object v2)
2343  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
      & v2 ~= null &
2344          sa..contents = sb..contents & sa..size = sb..size"
2345  modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2346  ensures "True" */
2347  {
2348    sa.remove(v1);
2349    /*: assume "~(v1 ~= v2)" */
2350    boolean r2a = sa.add(v2);
2351
2352    boolean r2b = sb.add(v2);
2353    sb.remove(v1);
2354
2355    /*: 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);

```

```

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(HashSet sa, HashSet sb, Object v1, Object v2)
2487 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
    & v2 ~= 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(HashSet sa, HashSet sb, Object v1, Object v2)
2503 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
    & v2 ~= 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(HashSet sa, HashSet sb, Object v1, Object
    v2)
2519 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
    & v2 ~= 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" */
2532 }
2533
2534 static void remove_contains_between_c_79(HashSet sa, HashSet sb, Object v1, Object
    v2)
2535 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
    & v2 ~= null &
2536         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
2552   & v2 ~= null &
2553     sa..contents = sb..contents & sa..size = sb..size"
2554   modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2555   ensures "True" */
2556 {
2557     sa.remove(v1);
2558     boolean r2a = sa.contains(v2);
2559     /*: assume "v1 ~= v2 | v1 ~: sa..(old contents)" */
2560
2561     boolean r2b = sb.contains(v2);
2562     sb.remove(v1);
2563
2564     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2565 }
2566
2567 static void remove_contains_post_c_80(HashSet sa, HashSet sb, Object v1, Object v2)
2568 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
2569   & v2 ~= null &
2570     sa..contents = sb..contents & sa..size = sb..size"
2571   modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2572   ensures "True" */
2573 {
2574     sa.remove(v1);
2575     boolean r2a = sa.contains(v2);
2576     /*: assume "~(v1 ~= v2 | v1 ~: sa..(old contents))" */
2577
2578     boolean r2b = sb.contains(v2);
2579     sb.remove(v1);
2580
2581     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2582 }
2583
2584 static void remove_remove_pre_s_81(HashSet sa, HashSet sb, Object v1, Object v2)
2585 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
2586   & v2 ~= null &
2587     sa..contents = sb..contents & sa..size = sb..size"
2588   modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2589   ensures "True" */
2590 {
2591     /*: assume "v1 ~= v2 | v1 ~: sa..contents" */
2592     sa.remove(v1);
2593     boolean r2a = sa.remove(v2);
2594
2595     boolean r2b = sb.remove(v2);
2596     sb.remove(v1);
2597
2598     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2599 }
2600
2601 static void remove_remove_pre_c_81(HashSet sa, HashSet sb, Object v1, Object v2)

```

```

2599  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
      & v2 ~= null &
2600          sa..contents = sb..contents & sa..size = sb..size"
2601  modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2602  ensures "True" */
2603  {
2604    /*: assume "~(v1 ~= v2 | v1 ~: sa..contents)" */
2605    sa.remove(v1);
2606    boolean r2a = sa.remove(v2);
2607
2608    boolean r2b = sb.remove(v2);
2609    sb.remove(v1);
2610
2611    /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2612  }
2613
2614  static void remove_remove_between_s_82(HashSet sa, HashSet sb, Object v1, Object v2)
2615  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
      & v2 ~= null &
2616          sa..contents = sb..contents & sa..size = sb..size"
2617  modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2618  ensures "True" */
2619  {
2620    sa.remove(v1);
2621    /*: assume "v1 ~= v2 | v1 ~: sa..(old contents)" */
2622    boolean r2a = sa.remove(v2);
2623
2624    boolean r2b = sb.remove(v2);
2625    sb.remove(v1);
2626
2627    /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2628  }
2629
2630  static void remove_remove_between_c_82(HashSet sa, HashSet sb, Object v1, Object v2)
2631  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
      & v2 ~= null &
2632          sa..contents = sb..contents & sa..size = sb..size"
2633  modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2634  ensures "True" */
2635  {
2636    sa.remove(v1);
2637    /*: assume "~(v1 ~= v2 | v1 ~: sa..(old contents))" */
2638    boolean r2a = sa.remove(v2);
2639
2640    boolean r2b = sb.remove(v2);
2641    sb.remove(v1);
2642
2643    /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2644  }
2645
2646  static void remove_remove_post_s_83(HashSet sa, HashSet sb, Object v1, Object v2)
2647  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
      & v2 ~= null &
2648          sa..contents = sb..contents & sa..size = sb..size"
2649  modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2650  ensures "True" */
2651  {
2652    sa.remove(v1);
2653    boolean r2a = sa.remove(v2);
2654    /*: assume "v1 ~= v2 | v1 ~: sa..(old contents)" */
2655
2656    boolean r2b = sb.remove(v2);
2657    sb.remove(v1);
2658
2659    /*: 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);

```

```

2721     sb.remove(v1);
2722
2723     /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
2724 }
2725
2726 static void remove_remove_between_c_85(HashSet sa, HashSet sb, Object v1, Object v2)
2727 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
    & v2 ~= null &
2728         sa..contents = sb..contents & sa..size = sb..size"
2729     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2730     ensures "True" */
2731 {
2732     sa.remove(v1);
2733     /*: assume "~(True)" */
2734     sa.remove(v2);
2735
2736     sb.remove(v2);
2737     sb.remove(v1);
2738
2739     /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
2740 }
2741
2742 static void remove_remove_post_s_86(HashSet sa, HashSet sb, Object v1, Object v2)
2743 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
    & v2 ~= null &
2744         sa..contents = sb..contents & sa..size = sb..size"
2745     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2746     ensures "True" */
2747 {
2748     sa.remove(v1);
2749     sa.remove(v2);
2750     /*: assume "True" */
2751
2752     sb.remove(v2);
2753     sb.remove(v1);
2754
2755     /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
2756 }
2757
2758 static void remove_remove_post_c_86(HashSet sa, HashSet sb, Object v1, Object v2)
2759 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
    & v2 ~= null &
2760         sa..contents = sb..contents & sa..size = sb..size"
2761     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2762     ensures "True" */
2763 {
2764     sa.remove(v1);
2765     sa.remove(v2);
2766     /*: assume "~(True)" */
2767
2768     sb.remove(v2);
2769     sb.remove(v1);
2770
2771     /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
2772 }
2773
2774 static void remove_size_pre_s_87(HashSet sa, HashSet sb, Object v1)
2775 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
    &
2776         sa..contents = sb..contents & sa..size = sb..size"
2777     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2778     ensures "True" */
2779 {
2780     /*: assume "v1 ~: sa..contents" */
2781     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     &
2793         sa..contents = sb..contents & sa..size = sb..size"
2794     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2795     ensures "True" */
2796     {
2797         /*: assume "~(v1 ~: sa..contents)" */
2798         sa.remove(v1);
2799         int r2a = sa.size();
2800
2801         int r2b = sb.size();
2802         sb.remove(v1);
2803
2804         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2805     }
2806
2807     static void remove_size_between_s_88(HashSet sa, HashSet sb, Object v1)
2808     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
2809     &
2810         sa..contents = sb..contents & sa..size = sb..size"
2811     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2812     ensures "True" */
2813     {
2814         sa.remove(v1);
2815         /*: assume "v1 ~: sa..(old contents)" */
2816         int r2a = sa.size();
2817
2818         int r2b = sb.size();
2819         sb.remove(v1);
2820
2821         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2822     }
2823
2824     static void remove_size_between_c_88(HashSet sa, HashSet sb, Object v1)
2825     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
2826     &
2827         sa..contents = sb..contents & sa..size = sb..size"
2828     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2829     ensures "True" */
2830     {
2831         sa.remove(v1);
2832         /*: assume "~(v1 ~: sa..(old contents))" */
2833         int r2a = sa.size();
2834
2835         int r2b = sb.size();
2836         sb.remove(v1);
2837
2838         /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2839     }
2840
2841     static void remove_size_post_s_89(HashSet sa, HashSet sb, Object v1)
2842     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v1 ~= null
2843     &
2844         sa..contents = sb..contents & sa..size = sb..size"
2845     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2846     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)

```

```

2903  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
      &
2904          sa..contents = sb..contents & sa..size = sb..size"
2905  modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2906  ensures "True" */
2907  {
2908      int r1a = sa.size();
2909      /*: assume "v2 : sa..contents" */
2910      boolean r2a = sa.add(v2);
2911
2912      boolean r2b = sb.add(v2);
2913      int r1b = sb.size();
2914
2915      /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
          sb..size" */
2916  }
2917
2918  static void size_add_between_c_91(HashSet sa, HashSet sb, Object v2)
2919  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashSet sa, HashSet sb, Object v2)
2935  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashSet sa, HashSet sb, Object v2)
2951  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashSet sa, HashSet sb, Object v2)
2967 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
    &
        sa..contents = sb..contents & sa..size = sb..size"
2968 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2969 ensures "True" */
2970 {
2971     /*: assume "v2 : sa..contents" */
2972     int r1a = sa.size();
2973     sa.add(v2);
2974
2975     sb.add(v2);
2976     int r1b = sb.size();
2977
2978     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
2979 }
2980
2981 static void size_add_pre_c_93(HashSet sa, HashSet sb, Object v2)
2982 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
    &
        sa..contents = sb..contents & sa..size = sb..size"
2983 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2984 ensures "True" */
2985 {
2986     /*: assume "~(v2 : sa..contents)" */
2987     int r1a = sa.size();
2988     sa.add(v2);
2989
2990     sb.add(v2);
2991     int r1b = sb.size();
2992
2993     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
2994 }
2995
2996 static void size_add_between_s_94(HashSet sa, HashSet sb, Object v2)
2997 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
    &
        sa..contents = sb..contents & sa..size = sb..size"
2998 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2999 ensures "True" */
3000 {
3001     int r1a = sa.size();
3002     /*: assume "v2 : sa..contents" */
3003     sa.add(v2);
3004
3005     sb.add(v2);
3006     int r1b = sb.size();
3007
3008     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
3009 }
3010
3011 static void size_add_between_c_94(HashSet sa, HashSet sb, Object v2)
3012 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
    &
        sa..contents = sb..contents & sa..size = sb..size"
3013 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3014 ensures "True" */
3015 {
3016     int r1a = sa.size();
3017
3018     sb.add(v2);
3019     int r1b = sb.size();
3020
3021     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..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 =
3092         sb..size)" */
3093 }
3094
3095 static void size_contains_between_s_97(HashSet sa, HashSet sb, Object v2)
3096 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
3097     &
3098         sa..contents = sb..contents & sa..size = sb..size"
3099     ensures "True" */
3100 {
3101     int r1a = sa.size();
3102     /*: assume "True" */
3103     boolean r2a = sa.contains(v2);
3104
3105     boolean r2b = sb.contains(v2);
3106     int r1b = sb.size();
3107
3108     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
3109         sb..size" */
3110 }
3111
3112 static void size_contains_between_c_97(HashSet sa, HashSet sb, Object v2)
3113 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
3114     &
3115         sa..contents = sb..contents & sa..size = sb..size"
3116     ensures "True" */
3117 {
3118     int r1a = sa.size();
3119     /*: assume "~(True)" */
3120     boolean r2a = sa.contains(v2);
3121
3122     boolean r2b = sb.contains(v2);
3123     int r1b = sb.size();
3124
3125     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
3126         sb..size)" */
3127 }
3128
3129 static void size_contains_post_s_98(HashSet sa, HashSet sb, Object v2)
3130 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & v2 ~= null
3131     &
3132         sa..contents = sb..contents & sa..size = sb..size"
3133     ensures "True" */
3134 {
3135     int r1a = sa.size();
3136     boolean r2a = sa.contains(v2);
3137     /*: assume "True" */
3138
3139     boolean r2b = sb.contains(v2);
3140     int r1b = sb.size();
3141
3142     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
3143         sb..size" */
3144 }
3145
3146
3147
3148

```

```

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);

```

```

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(HashSet sa, HashSet sb, Object v2)
3203 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashSet sa, HashSet sb, Object v2)
3219 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashSet sa, HashSet sb, Object v2)
3235 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashSet sa, HashSet sb, Object v2)
3251 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(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 =
3361          sb..size" */
3362  }
3363
3364  static void size_size_pre_c_105(HashSet sa, HashSet sb)
3365  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3366          sa..contents = sb..contents & sa..size = sb..size"
3367  ensures "True" */
3368  {
3369      /*: assume "~(True)" */
3370      int r1a = sa.size();
3371      int r2a = sa.size();
3372
3373      int r2b = sb.size();
3374      int r1b = sb.size();
3375
3376      /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
3377          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 =
3391           sb..size" */
3392 }
3393
3394 static void size_size_between_c_106(HashSet sa, HashSet sb)
3395 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3396           sa..contents = sb..contents & sa..size = sb..size"
3397     ensures "True" */
3398 {
3399     int r1a = sa.size();
3400     /*: assume "~(True)" */
3401     int r2a = sa.size();
3402
3403     int r2b = sb.size();
3404     int r1b = sb.size();
3405
3406     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
3407           sb..size)" */
3408 }
3409
3410 static void size_size_post_s_107(HashSet sa, HashSet sb)
3411 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3412           sa..contents = sb..contents & sa..size = sb..size"
3413     ensures "True" */
3414 {
3415     int r1a = sa.size();
3416     int r2a = sa.size();
3417     /*: assume "True" */
3418
3419     int r2b = sb.size();
3420     int r1b = sb.size();
3421
3422     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
3423           sb..size" */
3424 }
3425
3426 static void size_size_post_c_107(HashSet sa, HashSet sb)
3427 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3428           sa..contents = sb..contents & sa..size = sb..size"
3429     ensures "True" */
3430 {
3431     int r1a = sa.size();
3432     int r2a = sa.size();
3433     /*: assume "~(True)" */
3434
3435     int r2b = sb.size();
3436     int r1b = sb.size();
3437
3438     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
3439           sb..size)" */
3440 }

```

3437  
3438

}

### B.3.3 Inverse Testing Methods

Listing 9. HashSetInv.java

```
1 class HashSetInv {
2     static void add_0(HashSet s, Object v)
3     /*: requires "s ~= null & s..init & v ~= null"
4        modifies "s..contents", "s..size"
5        ensures "True" */
6     {
7         boolean r = s.add(v);
8         if (r) { s.remove(v); }
9
10        /*: assert "s..contents = s..(old contents) & s..size = s..(old size)" */
11    }
12
13    static void remove_1(HashSet s, Object v)
14    /*: requires "s ~= null & s..init & v ~= null"
15       modifies "s..contents", "s..size"
16       ensures "True" */
17    {
18        boolean r = s.remove(v);
19        if (r) { s.add(v); }
20
21        /*: assert "s..contents = s..(old contents) & s..size = s..(old size)" */
22    }
23
24 }
```

## B.4 AssociationList

### B.4.1 Data Structure

Listing 10. AssociationList.java

```
1 public /*: claimedby AssociationList */ class Node {
2     public Object key;
3     public Object value;
4     public Node next;
5
6     /*: public ghost specvar conts :: "(obj * obj) set" = "{}"
7        invariant CntDef: "ALL x. x : Node & x : alloc & x ~= null --> x..conts =
8           {(x..key, x..value)} Un x..next..conts & (ALL v. (x..key, v) ~:
9           x..next..conts)"
10       invariant CntNull: "null..conts = {}" */
11 }
12
13 public class AssociationList {
14     private Node head;
15     private int _size;
16
17     /*: public specvar contents :: "(obj * obj) set"
18        vardefs "contents == head..conts"
19        public ensured invariant MapInv: "ALL k v0 v1. (k, v0) : contents & (k, v1) :
20           contents --> v0 = v1"
21        invariant NonNullInv: "ALL k v. (k, v) : contents --> k ~= null & v ~= null"
22
23        static specvar edge :: "obj => obj => bool"
24        vardefs "edge == (%x y. (x : Node & y = x..next) | (x : AssociationList & y =
25           x..head))"
26        invariant InjInv: "ALL x1 x2 y. y ~= null & edge x1 y & edge x2 y --> x1 = x2"
27
28        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
44                     XIsThisHyp, LengthZero, ContentsEmpty */
45             }
46             {
47                 /*: assuming XNotThisHyp: "x ~= this" */
48                 /*: note XInOldAlloc: "x : old alloc" */
49                 /*: note OldCard: "x.(old AssociationList._size) = card (x.(old
50                     AssociationList.contents))" from CardHyp, XNotThisHyp,
51                     XInOldAlloc, CardInv */
52                 /*: note XLengthEq: "x._size = x.(old AssociationList._size)" */
53                 /*: note XContentsUnchanged: "x.contents = x.(old
54                     AssociationList.contents)" */
55                 /*: note XNotThisCard: "x._size = card (x.contents)" from
56                     XLengthEq, OldCard, XContentsUnchanged */
57             }
58             /*: note CardConc: "x._size = card (x.contents)" from XIsThisCard,
59                 XNotThisCard */
60         }
61         /*: note CardPostCond: "x : alloc & x : AssociationList --> x._size = card
62             (x.contents)" forSuch x */
63     }
64 }
65
66 public boolean containsKey(Object k0)
67 /*: ensures "result = (EX v. ((k0, v) : contents))" */
68 {
69     return _containsKey(k0);
70 }
71
72 private boolean _containsKey(Object k0)
73 /*: requires "theinvs"
74     ensures "result = (EX v. ((k0, v) : contents)) & theinvs" */
75 {
76     Node curr = head;
77     while /*: invariant "(EX v. (k0, v) : contents) = (EX v. (k0, v) :
78         curr..conts)" */ (curr != null) {
79         if (curr.key == k0) {
80             return true;
81         }
82         curr = curr.next;
83     }
84     return false;
85 }
86
87 private void _add(Object k0, Object v0)
88 /*: 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) :
```

```

128         old contents) & old contents = contents - {(k0, v0)} & old size = size - 1)"
129     */
130 {
131     if (_containsKey(k0)) {
132         Object v1 = _remove(k0);
133         _add(k0, v0);
134         return v1;
135     } else {
136         _add(k0, v0);
137         return null;
138     }
139 }
140
141 public Object get(Object k0)
142 /*: requires "k0 ~= null"
143 ensures "(EX v. (k0, v) : contents) --> (k0, result) : contents & result ~=
144 null) & ~(EX v. (k0, v) : contents) --> result = null)" */
145 {
146     Node curr = head;
147 while /*: invariant "ALL v. ((k0, v) : contents) = ((k0, v) : curr..conts)" */
148     (curr != null) {
149     if (curr.key == k0) {
150         return curr.value;
151     }
152     curr = curr.next;
153 }
154 return null;
155 }
156
157 public Object remove(Object k0)
158 /*: requires "k0 ~= null"
159 modifies "contents", "size"
160 ensures "(EX v. (k0, v) : old contents) --> (k0, result) : old contents &
161 contents = old contents - {(k0, result)} & size = old size - 1 & result ~=
162 null) & ~(EX v. (k0, v) : old contents) --> contents = old contents & size
163 = old size & result = null)" */
164 {
165     if (_containsKey(k0))
166         return _remove(k0);
167     else
168         return null;
169 }
170
171 private Object _remove(Object k0)
172 /*: requires "k0 ~= null & (EX v. (k0, v) : contents) & theinvs"
173 modifies "contents", "size", "head", "next", "conts", "_size"
174 ensures "(k0, result) : old contents & contents = old contents - {(k0, result)}
175 & size = old size - 1 & theinvs" */
176 {
177     /*: ghost specvar v0 :: obj */
178     /*: havoc v0 suchThat "(k0, v0) : contents" */
179     Node f = head;
180     if (f.key == k0) {
181         Node second = f.next;
182         f.next = null;
183         /*: "f..conts" := "{(f..key, f..value)}" */
184         head = second;
185         _size = _size - 1;
186         /*: note ContentsPost: "contents = old contents - {(f..key, f..value)}" */
187         {
188             /*: pickAny x :: obj */
189             {
190                 /*: assuming CardHyp: "x : alloc & x : AssociationList" */
191                 {

```

```

184      /*: note ThisProps: "this : old alloc & this : AssociationList"
185      */
186      /*: note OldCard: "old _size = card (old contents)" from
187      ThisProps, CardInv */
188      /*: note NewLength: "_size = old _size - 1" */
189      /*: note NewNotInOld: "(f..key, f..value) : old contents" */
190      /*: note XIsThisCard: "_size = card (contents)" from OldCard,
191      NewLength, NewNotInOld, ContentsPost */
192      }
193      {
194      /*: assuming XNotThisHyp: "x ~= this" */
195      /*: note XInOldAlloc: "x : old alloc" */
196      /*: note OldCard: "x..(old AssociationList._size) = card
197      (x..(old AssociationList.contents))" from CardHyp,
198      XNotThisHyp, XInOldAlloc, CardInv */
199      /*: note XLengthEq: "x.._size = x..(old AssociationList._size)"
200      */
201      {
202      /*: localize */
203      /*: note XContentsForw: "ALL y z. (y, z) : x..contents -->
204      (y, z) : x..(old AssociationList.contents)" */
205      /*: note XContentsBack: "ALL y z. (y, z) : x..(old
206      AssociationList.contents) --> (y, z) : x..contents" */
207      /*: note XContentsUnchanged: "x..contents = x..(old
208      AssociationList.contents)" from XContentsForw,
209      XContentsBack */
210      }
211      /*: note XNotThisCard: "x.._size = card (x..contents)" from
212      XLengthEq, OldCard, XContentsUnchanged */
213      }
214      /*: note CardConc: "x.._size = card (x..contents)" from
215      XIsThisCard, XNotThisCard */
216      }
217      /*: note CardPostCond: "x : alloc & x : AssociationList --> x.._size =
218      card (x..contents)" forSuch x */
219      }
220      return f.value;
221      } else {
222      Node prev = head;
223      /*: "prev..conts" := "prev..conts - {(k0, v0)}" */
224      Node curr = prev.next;
225      while /*: invariant "prev ~= null & prev..conts = prev..(old conts) - {(k0,
226      v0)} & curr ~= null & prev..next = curr & prev ~= curr & contents = old
227      contents - {(k0, v0)} & (ALL n. n : AssociationList & n : old alloc & n
228      ~= this --> n..contents = old (n..contents)) & (k0, v0) : curr..conts &
229      comment ''CntDefInv'' (ALL n. n : Node & n : alloc & n ~= null & n ~=
230      prev --> n..conts = {(n..key, n..value)} Un n..next..conts & (ALL v.
231      (n..key, v) ~: n..next..conts)) & (ALL n. n..conts = old (n..conts) |
232      n..conts = old (n..conts) - {(k0, v0)} & null..conts = {}" */ (curr.key
233      != k0) {
234      /*: "curr..conts" := "curr..conts - {(k0, v0)}" */
235      prev = curr;
236      curr = curr.next;
237      }
238      Node tmp = curr.next;
239      prev.next = tmp;
240      curr.next = null;
241      /*: "curr..conts" := "{(curr..key, curr..value)}" */
242      _size = _size - 1;
243      /*: note ContentsPost: "contents = old contents - {(curr..key,
244      curr..value)}" */
245      {
246      /*: pickAny x :: obj */
247      {
248      /*: assuming CardHyp: "x : alloc & x : AssociationList" */

```



```

227     {
228         /*: note ThisProps: "this : old alloc & this : AssociationList"
          */
229         /*: note OldCard: "old _size = card (old contents)" from
          ThisProps, CardInv */
230         /*: note NewLength: "_size = old _size - 1" */
231         /*: note NewNotInOld: "(curr..key, curr..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 AssociationList._size) = card
          (x..(old AssociationList.contents))" from CardHyp,
          XNotThisHyp, XInOldAlloc, CardInv */
238         /*: note XLengthEq: "x.._size = x..(old AssociationList._size)"
          */
239         {
240             /*: localize */
241             /*: note XContentsForw: "ALL y z. (y, z) : x..contents -->
          (y, z) : x..(old AssociationList.contents)" */
242             /*: note XContentsBack: "ALL y z. (y, z) : x..(old
          AssociationList.contents) --> (y, z) : x..contents" */
243             /*: note XContentsUnchanged: "x..contents = x..(old
          AssociationList.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 : AssociationList --> x.._size =
          card (x..contents)" forSuch x */
250 }
251     return curr.value;
252 }
253 }
254
255 public int size()
256 /*: ensures "result = size" */
257 {
258     return _size;
259 }
260 }

```

## B.4.2 Commutativity Testing Methods

Listing 11. AssociationListComm.java

```

1 class AssociationListComm {
2     static void containsKey_containsKey_pre_s_0(AssociationList sa, AssociationList sb,
3         Object k1, Object k2)
4     /*: requires "sa ~= null & sb ~= null & sa ~= sb &
5         sa..contents = sb..contents & sa..size = sb..size"
6         ensures "True" */
7     {
8         /*: assume "True" */
9         boolean r1a = sa.containsKey(k1);
10        boolean r2a = sa.containsKey(k2);
11
12        boolean r2b = sb.containsKey(k2);

```

```

12     boolean r1b = sb.containsKey(k1);
13
14     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
15         sb..size" */
16 }
17
18 static void containsKey_containsKey_pre_c_0(AssociationList sa, AssociationList sb,
19     Object k1, Object k2)
20 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
21     sa..contents = sb..contents & sa..size = sb..size"
22     ensures "True" */
23 {
24     /*: assume "~(True)" */
25     boolean r1a = sa.containsKey(k1);
26     boolean r2a = sa.containsKey(k2);
27
28     boolean r2b = sb.containsKey(k2);
29     boolean r1b = sb.containsKey(k1);
30
31     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
32         sb..size)" */
33 }
34
35 static void containsKey_containsKey_between_s_1(AssociationList sa, AssociationList
36     sb, Object k1, Object k2)
37 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
38     sa..contents = sb..contents & sa..size = sb..size"
39     ensures "True" */
40 {
41     boolean r1a = sa.containsKey(k1);
42     /*: assume "True" */
43     boolean r2a = sa.containsKey(k2);
44
45     boolean r2b = sb.containsKey(k2);
46     boolean r1b = sb.containsKey(k1);
47
48     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
49         sb..size" */
50 }
51
52 static void containsKey_containsKey_between_c_1(AssociationList sa, AssociationList
53     sb, Object k1, Object k2)
54 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
55     sa..contents = sb..contents & sa..size = sb..size"
56     ensures "True" */
57 {
58     boolean r1a = sa.containsKey(k1);
59     /*: assume "~(True)" */
60     boolean r2a = sa.containsKey(k2);
61
62     boolean r2b = sb.containsKey(k2);
63     boolean r1b = sb.containsKey(k1);
64
65     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
66         sb..size)" */
67 }
68
69 static void containsKey_containsKey_post_s_2(AssociationList sa, AssociationList
70     sb, Object k1, Object k2)
71 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
72     sa..contents = sb..contents & sa..size = sb..size"
73     ensures "True" */
74 {
75     boolean r1a = sa.containsKey(k1);
76     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 =
75         sb..size" */
76 }
77
78 static void containsKey_containsKey_post_c_2(AssociationList sa, AssociationList
79     sb, Object k1, Object k2)
80 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
81     sa..contents = sb..contents & sa..size = sb..size"
82     ensures "True" */
83 {
84     boolean r1a = sa.containsKey(k1);
85     boolean r2a = sa.containsKey(k2);
86     /*: assume "~(True)" */
87
88     boolean r2b = sb.containsKey(k2);
89     boolean r1b = sb.containsKey(k1);
90
91     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
92         sb..size)" */
93 }
94
95 static void containsKey_get_pre_s_3(AssociationList sa, AssociationList sb, Object
96     k1, Object k2)
97 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
98     sa..contents = sb..contents & sa..size = sb..size"
99     ensures "True" */
100 {
101     /*: assume "True" */
102     boolean r1a = sa.containsKey(k1);
103     Object r2a = sa.get(k2);
104
105     Object r2b = sb.get(k2);
106     boolean r1b = sb.containsKey(k1);
107
108     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
109         sb..size" */
110 }
111
112 static void containsKey_get_pre_c_3(AssociationList sa, AssociationList sb, Object
113     k1, Object k2)
114 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
115     sa..contents = sb..contents & sa..size = sb..size"
116     ensures "True" */
117 {
118     /*: assume "~(True)" */
119     boolean r1a = sa.containsKey(k1);
120     Object r2a = sa.get(k2);
121
122     Object r2b = sb.get(k2);
123     boolean r1b = sb.containsKey(k1);
124
125     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
126         sb..size)" */
127 }
128
129 static void containsKey_get_between_s_4(AssociationList sa, AssociationList sb,
130     Object k1, Object k2)
131 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
132     sa..contents = sb..contents & sa..size = sb..size"
133     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 =
135         sb..size" */
136 }
137
138 static void containsKey_get_between_c_4(AssociationList sa, AssociationList sb,
139     Object k1, Object k2)
140 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
141     sa..contents = sb..contents & sa..size = sb..size"
142     ensures "True" */
143 {
144     boolean r1a = sa.containsKey(k1);
145     /*: assume "~(True)" */
146     Object r2a = sa.get(k2);
147
148     Object r2b = sb.get(k2);
149     boolean r1b = sb.containsKey(k1);
150
151     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
152         sb..size)" */
153 }
154
155 static void containsKey_get_post_s_5(AssociationList sa, AssociationList sb, Object
156     k1, Object k2)
157 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
158     sa..contents = sb..contents & sa..size = sb..size"
159     ensures "True" */
160 {
161     boolean r1a = sa.containsKey(k1);
162     Object r2a = sa.get(k2);
163     /*: assume "True" */
164
165     Object r2b = sb.get(k2);
166     boolean r1b = sb.containsKey(k1);
167
168     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
169         sb..size" */
170 }
171
172 static void containsKey_get_post_c_5(AssociationList sa, AssociationList sb, Object
173     k1, Object k2)
174 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
175     sa..contents = sb..contents & sa..size = sb..size"
176     ensures "True" */
177 {
178     boolean r1a = sa.containsKey(k1);
179     Object r2a = sa.get(k2);
180     /*: assume "~(True)" */
181
182     Object r2b = sb.get(k2);
183     boolean r1b = sb.containsKey(k1);
184
185     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
186         sb..size)" */
187 }
188
189 static void containsKey_put_pre_s_6(AssociationList sa, AssociationList sb, Object
190     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 =
196          sb..size" */
197  }
198
199  static void containsKey_put_pre_c_6(AssociationList sa, AssociationList sb, Object
200      k1, Object k2, Object v2)
201  /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
202      sa..contents = sb..contents & sa..size = sb..size"
203  modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
204  ensures "True" */
205  {
206      /*: assume "~(k1 ~= k2 | (EX v. (k1, v) : sa..contents))" */
207      boolean r1a = sa.containsKey(k1);
208      Object r2a = sa.put(k2, v2);
209
210      Object r2b = sb.put(k2, v2);
211      boolean r1b = sb.containsKey(k1);
212
213      /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
214          sb..size)" */
215  }
216
217  static void containsKey_put_between_s_7(AssociationList sa, AssociationList sb,
218      Object k1, Object k2, Object v2)
219  /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
220      sa..contents = sb..contents & sa..size = sb..size"
221  modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
222  ensures "True" */
223  {
224      boolean r1a = sa.containsKey(k1);
225      /*: assume "k1 ~= k2 | r1a" */
226      Object r2a = sa.put(k2, v2);
227
228      Object r2b = sb.put(k2, v2);
229      boolean r1b = sb.containsKey(k1);
230
231      /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
232          sb..size" */
233  }
234
235  static void containsKey_put_between_c_7(AssociationList sa, AssociationList sb,
236      Object k1, Object k2, Object v2)
237  /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
238      sa..contents = sb..contents & sa..size = sb..size"
239  modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
240  ensures "True" */
241  {
242      boolean r1a = sa.containsKey(k1);
243      /*: assume "~(k1 ~= k2 | r1a)" */
244      Object r2a = sa.put(k2, v2);
245
246      Object r2b = sb.put(k2, v2);
247      boolean r1b = sb.containsKey(k1);

```

```

242     /*: assert "~(ria = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
243     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 "ria = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
260     sb..size" */
261 }
262
263 static void containsKey_put_post_c_8(AssociationList sa, AssociationList sb, Object
    k1, Object k2, Object v2)
264 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
265     sa..contents = sb..contents & sa..size = sb..size"
266     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
267     ensures "True" */
268 {
269     boolean r1a = sa.containsKey(k1);
270     Object r2a = sa.put(k2, v2);
271     /*: assume "~(k1 ~= k2 | r1a)" */
272
273     Object r2b = sb.put(k2, v2);
274     boolean r1b = sb.containsKey(k1);
275
276     /*: assert "~(ria = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
277     sb..size)" */
278 }
279
280 static void containsKey_put_pre_s_9(AssociationList sa, AssociationList sb, Object
    k1, Object k2, Object v2)
281 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
282     sa..contents = sb..contents & sa..size = sb..size"
283     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
284     ensures "True" */
285 {
286     /*: assume "k1 ~= k2 | (EX v. (k1, v) : sa..contents)" */
287     boolean r1a = sa.containsKey(k1);
288     sa.put(k2, v2);
289
290     sb.put(k2, v2);
291     boolean r1b = sb.containsKey(k1);
292
293     /*: assert "ria = r1b & sa..contents = sb..contents & sa..size = sb..size" */
294 }
295
296 static void containsKey_put_pre_c_9(AssociationList sa, AssociationList sb, Object
    k1, Object k2, Object v2)
297 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
298     sa..contents = sb..contents & sa..size = sb..size"
299     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
300     ensures "True" */
301 {

```

```

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(AssociationList sa, AssociationList sb,
311     Object k1, Object k2, Object v2)
312 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
313     sa..contents = sb..contents & sa..size = sb..size"
314     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
315     ensures "True" */
316 {
317     boolean r1a = sa.containsKey(k1);
318     /*: assume "k1 ~= k2 | r1a" */
319     sa.put(k2, v2);
320
321     sb.put(k2, v2);
322     boolean r1b = sb.containsKey(k1);
323
324     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
325 }
326
327 static void containsKey_put_between_c_10(AssociationList sa, AssociationList sb,
328     Object k1, Object k2, Object v2)
329 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
330     sa..contents = sb..contents & sa..size = sb..size"
331     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
332     ensures "True" */
333 {
334     boolean r1a = sa.containsKey(k1);
335     /*: assume "~(k1 ~= k2 | r1a)" */
336     sa.put(k2, v2);
337
338     sb.put(k2, v2);
339     boolean r1b = sb.containsKey(k1);
340
341     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
342 }
343
344 static void containsKey_put_post_s_11(AssociationList sa, AssociationList sb,
345     Object k1, Object k2, Object v2)
346 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
347     sa..contents = sb..contents & sa..size = sb..size"
348     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
349     ensures "True" */
350 {
351     boolean r1a = sa.containsKey(k1);
352     sa.put(k2, v2);
353     /*: assume "k1 ~= k2 | r1a" */
354
355     sb.put(k2, v2);
356     boolean r1b = sb.containsKey(k1);
357
358     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
359 }
360
361 static void containsKey_put_post_c_11(AssociationList sa, AssociationList sb,
362     Object k1, Object k2, Object v2)
363 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
364     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,
375     Object k1, Object k2)
376 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
377     sa..contents = sb..contents & sa..size = sb..size"
378     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
379     ensures "True" */
380 {
381     /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
382     boolean r1a = sa.containsKey(k1);
383     Object r2a = sa.remove(k2);
384
385     Object r2b = sb.remove(k2);
386     boolean r1b = sb.containsKey(k1);
387
388     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
389     sb..size" */
390 }
391
392 static void containsKey_remove_pre_c_12(AssociationList sa, AssociationList sb,
393     Object k1, Object k2)
394 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
395     sa..contents = sb..contents & sa..size = sb..size"
396     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
397     ensures "True" */
398 {
399     /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
400     boolean r1a = sa.containsKey(k1);
401     Object r2a = sa.remove(k2);
402
403     Object r2b = sb.remove(k2);
404     boolean r1b = sb.containsKey(k1);
405
406     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
407     sb..size)" */
408 }
409
410 static void containsKey_remove_between_s_13(AssociationList sa, AssociationList sb,
411     Object k1, Object k2)
412 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
413     sa..contents = sb..contents & sa..size = sb..size"
414     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
415     ensures "True" */
416 {
417     boolean r1a = sa.containsKey(k1);
418     /*: assume "k1 ~= k2 | ~r1a" */
419     Object r2a = sa.remove(k2);
420
421     Object r2b = sb.remove(k2);
422     boolean r1b = sb.containsKey(k1);
423
424     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
425     sb..size" */

```



```

420 }
421
422 static void containsKey_remove_between_c_13(AssociationList sa, AssociationList sb,
423      Object k1, Object k2)
424 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
425      sa..contents = sb..contents & sa..size = sb..size"
426      modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
427      ensures "True" */
428 {
429     boolean r1a = sa.containsKey(k1);
430     /*: assume "~(k1 ~= k2 | ~r1a)" */
431     Object r2a = sa.remove(k2);
432
433     Object r2b = sb.remove(k2);
434     boolean r1b = sb.containsKey(k1);
435
436     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
437      sb..size)" */
438 }
439
440 static void containsKey_remove_post_s_14(AssociationList sa, AssociationList sb,
441      Object k1, Object k2)
442 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
443      sa..contents = sb..contents & sa..size = sb..size"
444      modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
445      ensures "True" */
446 {
447     boolean r1a = sa.containsKey(k1);
448     Object r2a = sa.remove(k2);
449     /*: assume "k1 ~= k2 | ~r1a" */
450
451     Object r2b = sb.remove(k2);
452     boolean r1b = sb.containsKey(k1);
453
454     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
455      sb..size" */
456 }
457
458 static void containsKey_remove_post_c_14(AssociationList sa, AssociationList sb,
459      Object k1, Object k2)
460 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
461      sa..contents = sb..contents & sa..size = sb..size"
462      modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
463      ensures "True" */
464 {
465     boolean r1a = sa.containsKey(k1);
466     Object r2a = sa.remove(k2);
467     /*: assume "~(k1 ~= k2 | ~r1a)" */
468
469     Object r2b = sb.remove(k2);
470     boolean r1b = sb.containsKey(k1);
471
472     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
473      sb..size)" */
474 }
475
476 static void containsKey_remove_pre_s_15(AssociationList sa, AssociationList sb,
477      Object k1, Object k2)
478 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
479      sa..contents = sb..contents & sa..size = sb..size"
480      modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
481      ensures "True" */
482 {
483     /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
484     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,
551     Object k1, Object k2)
552 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
553     sa..contents = sb..contents & sa..size = sb..size"
554     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
555     ensures "True" */
556 {
557     boolean r1a = sa.containsKey(k1);
558     sa.remove(k2);
559     /*: assume "~(k1 ~= k2 | ~r1a)" */
560
561     sb.remove(k2);
562     boolean r1b = sb.containsKey(k1);
563
564     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
565 }
566
567 static void containsKey_size_pre_s_18(AssociationList sa, AssociationList sb,
568     Object k1)
569 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
570     sa..contents = sb..contents & sa..size = sb..size"
571     ensures "True" */
572 {
573     /*: assume "True" */
574     boolean r1a = sa.containsKey(k1);
575     int r2a = sa.size();
576
577     int r2b = sb.size();
578     boolean r1b = sb.containsKey(k1);
579
580     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
581     sb..size" */
582 }
583
584 static void containsKey_size_pre_c_18(AssociationList sa, AssociationList sb,
585     Object k1)
586 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
587     sa..contents = sb..contents & sa..size = sb..size"
588     ensures "True" */
589 {
590     /*: assume "~(True)" */
591     boolean r1a = sa.containsKey(k1);
592     int r2a = sa.size();
593
594     int r2b = sb.size();
595     boolean r1b = sb.containsKey(k1);
596
597     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
598     sb..size)" */
599 }
600
601 static void containsKey_size_between_s_19(AssociationList sa, AssociationList sb,
602     Object k1)
603 /*: 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 &
        sa..contents = sb..contents & sa..size = sb..size"
613     ensures "True" */
614 {
615     boolean r1a = sa.containsKey(k1);
616     /*: assume "~(True)" */
617     int r2a = sa.size();
618
619     int r2b = sb.size();
620     boolean r1b = sb.containsKey(k1);
621
622     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
        sb..size)" */
623 }
624
625 static void containsKey_size_post_s_20(AssociationList sa, AssociationList sb,
        Object k1)
626 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
        sa..contents = sb..contents & sa..size = sb..size"
627     ensures "True" */
628 {
629     boolean r1a = sa.containsKey(k1);
630     int r2a = sa.size();
631     /*: assume "True" */
632
633     int r2b = sb.size();
634     boolean r1b = sb.containsKey(k1);
635
636     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
        sb..size" */
637 }
638
639 static void containsKey_size_post_c_20(AssociationList sa, AssociationList sb,
        Object k1)
640 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
        sa..contents = sb..contents & sa..size = sb..size"
641     ensures "True" */
642 {
643     boolean r1a = sa.containsKey(k1);
644     int r2a = sa.size();
645     /*: assume "~(True)" */
646
647     int r2b = sb.size();
648     boolean r1b = sb.containsKey(k1);
649
650     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
        sb..size)" */
651 }
652
653
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 =
714         sb..size)" */
715 }
716 static void get_containsKey_post_s_23(AssociationList sa, AssociationList sb,
717     Object k1, Object k2)
718 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
719     sa..contents = sb..contents & sa..size = sb..size"
720     ensures "True" */
721 {
722     Object r1a = sa.get(k1);
723     boolean r2a = sa.containsKey(k2);
724     /*: assume "True" */
725
726     boolean r2b = sb.containsKey(k2);
727     Object r1b = sb.get(k1);
728
729     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
730         sb..size" */
731 }
732 static void get_containsKey_post_c_23(AssociationList sa, AssociationList sb,
733     Object k1, Object k2)
734 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
735     sa..contents = sb..contents & sa..size = sb..size"
736     ensures "True" */
737 {
738     Object r1a = sa.get(k1);
739     boolean r2a = sa.containsKey(k2);
740     /*: assume "~(True)" */
741
742     boolean r2b = sb.containsKey(k2);
743     Object r1b = sb.get(k1);
744
745     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
746         sb..size)" */
747 }
748 static void get_get_pre_s_24(AssociationList sa, AssociationList sb, Object k1,
749     Object k2)
750 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
751     sa..contents = sb..contents & sa..size = sb..size"
752     ensures "True" */
753 {
754     /*: assume "True" */
755     Object r1a = sa.get(k1);
756     Object r2a = sa.get(k2);
757
758     Object r2b = sb.get(k2);
759     Object r1b = sb.get(k1);
760
761     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
762         sb..size" */
763 }
764 static void get_get_pre_c_24(AssociationList sa, AssociationList sb, Object k1,
765     Object k2)
766 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
767     sa..contents = sb..contents & sa..size = sb..size"
768     ensures "True" */
769 {
770     /*: assume "~(True)" */
771     Object r1a = sa.get(k1);
772     Object r2a = sa.get(k2);

```

```

770     Object r2b = sb.get(k2);
771     Object r1b = sb.get(k1);
772
773     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
774         sb..size)" */
775 }
776
777 static void get_get_between_s_25(AssociationList sa, AssociationList sb, Object k1,
778     Object k2)
779 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
780     sa..contents = sb..contents & sa..size = sb..size"
781     ensures "True" */
782 {
783     Object r1a = sa.get(k1);
784     /*: assume "True" */
785     Object r2a = sa.get(k2);
786
787     Object r2b = sb.get(k2);
788     Object r1b = sb.get(k1);
789
790     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
791         sb..size" */
792 }
793
794 static void get_get_between_c_25(AssociationList sa, AssociationList sb, Object k1,
795     Object k2)
796 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
797     sa..contents = sb..contents & sa..size = sb..size"
798     ensures "True" */
799 {
800     Object r1a = sa.get(k1);
801     /*: assume "~(True)" */
802     Object r2a = sa.get(k2);
803
804     Object r2b = sb.get(k2);
805     Object r1b = sb.get(k1);
806
807     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
808         sb..size)" */
809 }
810
811 static void get_get_post_s_26(AssociationList sa, AssociationList sb, Object k1,
812     Object k2)
813 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
814     sa..contents = sb..contents & sa..size = sb..size"
815     ensures "True" */
816 {
817     Object r1a = sa.get(k1);
818     Object r2a = sa.get(k2);
819     /*: assume "True" */
820
821     Object r2b = sb.get(k2);
822     Object r1b = sb.get(k1);
823
824     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
825         sb..size" */
826 }
827
828 static void get_get_post_c_26(AssociationList sa, AssociationList sb, Object k1,
829     Object k2)
830 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
831     sa..contents = sb..contents & sa..size = sb..size"
832     ensures "True" */
833 {
834     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,
885     Object k2, Object v2)
886 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
887     null &
888     sa..contents = sb..contents & sa..size = sb..size"
889     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
890     ensures "True" */
891 {
892     Object r1a = sa.get(k1);
893     /*: assume "~(k1 ~= k2 | r1a = v2)" */
894     Object r2a = sa.put(k2, v2);
895
896     Object r2b = sb.put(k2, v2);
897     Object r1b = sb.get(k1);
898
899     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
900     sb..size)" */
901 }
902
903 static void get_put_post_s_29(AssociationList sa, AssociationList sb, Object k1,
904     Object k2, Object v2)
905 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
906     null &
907     sa..contents = sb..contents & sa..size = sb..size"
908     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
909     ensures "True" */
910 {
911     Object r1a = sa.get(k1);
912     Object r2a = sa.put(k2, v2);
913     /*: assume "k1 ~= k2 | r1a = v2" */
914
915     Object r2b = sb.put(k2, v2);
916     Object r1b = sb.get(k1);
917
918     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
919     sb..size" */
920 }
921
922 static void get_put_post_c_29(AssociationList sa, AssociationList sb, Object k1,
923     Object k2, Object v2)
924 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
925     null &
926     sa..contents = sb..contents & sa..size = sb..size"
927     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
928     ensures "True" */
929 {
930     Object r1a = sa.get(k1);
931     Object r2a = sa.put(k2, v2);
932     /*: assume "~(k1 ~= k2 | r1a = v2)" */
933
934     Object r2b = sb.put(k2, v2);
935     Object r1b = sb.get(k1);
936
937     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
938     sb..size)" */
939 }
940
941 static void get_put_pre_s_30(AssociationList sa, AssociationList sb, Object k1,
942     Object k2, Object v2)
943 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
944     null &
945     sa..contents = sb..contents & sa..size = sb..size"
946     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,
949     Object k2, Object v2)
950 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
951     null &
952     sa..contents = sb..contents & sa..size = sb..size"
953     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
954     ensures "True" */
955 {
956     /*: assume "~(k1 ~= k2 | (k1, v2) : sa..contents)" */
957     Object r1a = sa.get(k1);
958     sa.put(k2, v2);
959
960     sb.put(k2, v2);
961     Object r1b = sb.get(k1);
962
963     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
964 }
965
966 static void get_put_between_s_31(AssociationList sa, AssociationList sb, Object k1,
967     Object k2, Object v2)
968 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
969     null &
970     sa..contents = sb..contents & sa..size = sb..size"
971     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
972     ensures "True" */
973 {
974     Object r1a = sa.get(k1);
975     /*: assume "k1 ~= k2 | r1a = v2" */
976     sa.put(k2, v2);
977
978     sb.put(k2, v2);
979     Object r1b = sb.get(k1);
980
981     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
982 }
983
984 static void get_put_between_c_31(AssociationList sa, AssociationList sb, Object k1,
985     Object k2, Object v2)
986 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
987     null &
988     sa..contents = sb..contents & sa..size = sb..size"
989     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
990     ensures "True" */
991 {
992     Object r1a = sa.get(k1);
993     /*: assume "~(k1 ~= k2 | r1a = v2)" */
994     sa.put(k2, v2);
995
996     sb.put(k2, v2);
997     Object r1b = sb.get(k1);
998
999     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1000 }

```

```

995 static void get_put_post_s_32(AssociationList sa, AssociationList sb, Object k1,
996     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,
1013     Object k2, Object v2)
1014 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
    null &
1015     sa..contents = sb..contents & sa..size = sb..size"
1016     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1017     ensures "True" */
1018 {
1019     Object r1a = sa.get(k1);
1020     sa.put(k2, v2);
1021     /*: assume "~(k1 ~= k2 | r1a = v2)" */
1022
1023     sb.put(k2, v2);
1024     Object r1b = sb.get(k1);
1025
1026     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1027 }
1028
1029 static void get_remove_pre_s_33(AssociationList sa, AssociationList sb, Object k1,
1030     Object k2)
1031 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
    sa..contents = sb..contents & sa..size = sb..size"
1032     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1033     ensures "True" */
1034 {
1035     /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
1036     Object r1a = sa.get(k1);
1037     Object r2a = sa.remove(k2);
1038
1039     Object r2b = sb.remove(k2);
1040     Object r1b = sb.get(k1);
1041
1042     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
    sb..size" */
1043 }
1044
1045 static void get_remove_pre_c_33(AssociationList sa, AssociationList sb, Object k1,
1046     Object k2)
1047 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
    sa..contents = sb..contents & sa..size = sb..size"
1048     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1049     ensures "True" */
1050 {
1051     /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
1052     Object r1a = sa.get(k1);
1053     Object r2a = sa.remove(k2);

```

```

1053     Object r2b = sb.remove(k2);
1054     Object r1b = sb.get(k1);
1055
1056     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1057         sb..size)" */
1058 }
1059
1060 static void get_remove_between_s_34(AssociationList sa, AssociationList sb, Object
1061     k1, Object k2)
1062 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
1063     sa..contents = sb..contents & sa..size = sb..size"
1064     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1065     ensures "True" */
1066 {
1067     Object r1a = sa.get(k1);
1068     /*: assume "k1 ~= k2 | r1a = null" */
1069     Object r2a = sa.remove(k2);
1070
1071     Object r2b = sb.remove(k2);
1072     Object r1b = sb.get(k1);
1073
1074     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1075         sb..size" */
1076 }
1077
1078 static void get_remove_between_c_34(AssociationList sa, AssociationList sb, Object
1079     k1, Object k2)
1080 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
1081     sa..contents = sb..contents & sa..size = sb..size"
1082     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1083     ensures "True" */
1084 {
1085     Object r1a = sa.get(k1);
1086     /*: assume "~(k1 ~= k2 | r1a = null)" */
1087     Object r2a = sa.remove(k2);
1088
1089     Object r2b = sb.remove(k2);
1090     Object r1b = sb.get(k1);
1091
1092     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1093         sb..size)" */
1094 }
1095
1096 static void get_remove_post_s_35(AssociationList sa, AssociationList sb, Object k1,
1097     Object k2)
1098 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
1099     sa..contents = sb..contents & sa..size = sb..size"
1100     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1101     ensures "True" */
1102 {
1103     Object r1a = sa.get(k1);
1104     Object r2a = sa.remove(k2);
1105     /*: assume "k1 ~= k2 | r1a = null" */
1106
1107     Object r2b = sb.remove(k2);
1108     Object r1b = sb.get(k1);
1109
1110     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1111         sb..size" */
1112 }
1113
1114 static void get_remove_post_c_35(AssociationList sa, AssociationList sb, Object k1,
1115     Object k2)
1116 /*: 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 =
1122         sb..size)" */
1123 }
1124
1125 static void get_remove_pre_s_36(AssociationList sa, AssociationList sb, Object k1,
1126     Object k2)
1127 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
1128     sa..contents = sb..contents & sa..size = sb..size"
1129     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1130     ensures "True" */
1131 {
1132     /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
1133     Object r1a = sa.get(k1);
1134     sa.remove(k2);
1135
1136     sb.remove(k2);
1137     Object r1b = sb.get(k1);
1138
1139     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1140 }
1141
1142 static void get_remove_pre_c_36(AssociationList sa, AssociationList sb, Object k1,
1143     Object k2)
1144 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
1145     sa..contents = sb..contents & sa..size = sb..size"
1146     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1147     ensures "True" */
1148 {
1149     /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
1150     Object r1a = sa.get(k1);
1151     sa.remove(k2);
1152
1153     sb.remove(k2);
1154     Object r1b = sb.get(k1);
1155
1156     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1157 }
1158
1159 static void get_remove_between_s_37(AssociationList sa, AssociationList sb, Object
1160     k1, Object k2)
1161 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
1162     sa..contents = sb..contents & sa..size = sb..size"
1163     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1164     ensures "True" */
1165 {
1166     Object r1a = sa.get(k1);
1167     /*: assume "k1 ~= k2 | r1a = null" */
1168     sa.remove(k2);
1169
1170     sb.remove(k2);
1171     Object r1b = sb.get(k1);
1172
1173     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1174 }

```

```

1171 static void get_remove_between_c_37(AssociationList sa, AssociationList sb, Object
1172 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,
1189 Object k2)
1190 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
1191 sa..contents = sb..contents & sa..size = sb..size"
1192 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1193 ensures "True" */
1194 {
1195 Object r1a = sa.get(k1);
1196 sa.remove(k2);
1197 /*: assume "k1 ~= k2 | r1a = null" */
1198
1199 sb.remove(k2);
1200 Object r1b = sb.get(k1);
1201
1202 /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1203 }
1204
1205 static void get_remove_post_c_38(AssociationList sa, AssociationList sb, Object k1,
1206 Object k2)
1207 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
1208 sa..contents = sb..contents & sa..size = sb..size"
1209 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1210 ensures "True" */
1211 {
1212 Object r1a = sa.get(k1);
1213 sa.remove(k2);
1214 /*: assume "~(k1 ~= k2 | r1a = null)" */
1215
1216 sb.remove(k2);
1217 Object r1b = sb.get(k1);
1218
1219 /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1220 }
1221
1222 static void get_size_pre_s_39(AssociationList sa, AssociationList sb, Object k1)
1223 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
1224 sa..contents = sb..contents & sa..size = sb..size"
1225 ensures "True" */
1226 {
1227 /*: assume "True" */
1228 Object r1a = sa.get(k1);
1229 int r2a = sa.size();
1230
1231 int r2b = sb.size();
1232 Object r1b = sb.get(k1);
1233 }

```

```

1232     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1233         sb..size" */
1234 }
1235
1236 static void get_size_pre_c_39(AssociationList sa, AssociationList sb, Object k1)
1237 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
1238     sa..contents = sb..contents & sa..size = sb..size"
1239     ensures "True" */
1240 {
1241     /*: assume "~(True)" */
1242     Object r1a = sa.get(k1);
1243     int r2a = sa.size();
1244
1245     int r2b = sb.size();
1246     Object r1b = sb.get(k1);
1247
1248     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1249         sb..size)" */
1250 }
1251
1252 static void get_size_between_s_40(AssociationList sa, AssociationList sb, Object k1)
1253 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
1254     sa..contents = sb..contents & sa..size = sb..size"
1255     ensures "True" */
1256 {
1257     Object r1a = sa.get(k1);
1258     /*: assume "True" */
1259     int r2a = sa.size();
1260
1261     int r2b = sb.size();
1262     Object r1b = sb.get(k1);
1263
1264     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1265         sb..size" */
1266 }
1267
1268 static void get_size_between_c_40(AssociationList sa, AssociationList sb, Object k1)
1269 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
1270     sa..contents = sb..contents & sa..size = sb..size"
1271     ensures "True" */
1272 {
1273     Object r1a = sa.get(k1);
1274     /*: assume "~(True)" */
1275     int r2a = sa.size();
1276
1277     int r2b = sb.size();
1278     Object r1b = sb.get(k1);
1279
1280     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1281         sb..size)" */
1282 }
1283
1284 static void get_size_post_s_41(AssociationList sa, AssociationList sb, Object k1)
1285 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
1286     sa..contents = sb..contents & sa..size = sb..size"
1287     ensures "True" */
1288 {
1289     Object r1a = sa.get(k1);
1290     int r2a = sa.size();
1291     /*: assume "True" */
1292
1293     int r2b = sb.size();
1294     Object r1b = sb.get(k1);

```

```

1292     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1293         sb..size" */
1294 }
1295 static void get_size_post_c_41(AssociationList sa, AssociationList sb, Object k1)
1296 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
1297     sa..contents = sb..contents & sa..size = sb..size"
1298     ensures "True" */
1299 {
1300     Object r1a = sa.get(k1);
1301     int r2a = sa.size();
1302     /*: assume "~(True)" */
1303
1304     int r2b = sb.size();
1305     Object r1b = sb.get(k1);
1306
1307     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1308         sb..size)" */
1309 }
1310 static void put_containsKey_pre_s_42(AssociationList sa, AssociationList sb, Object
1311     k1, Object v1, Object k2)
1312 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
1313     sa..contents = sb..contents & sa..size = sb..size"
1314     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1315     ensures "True" */
1316 {
1317     /*: assume "k1 ~= k2 | (EX v. (k1, v) : sa..contents)" */
1318     Object r1a = sa.put(k1, v1);
1319     boolean r2a = sa.containsKey(k2);
1320
1321     boolean r2b = sb.containsKey(k2);
1322     Object r1b = sb.put(k1, v1);
1323
1324     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1325         sb..size" */
1326 }
1327 static void put_containsKey_pre_c_42(AssociationList sa, AssociationList sb, Object
1328     k1, Object v1, Object k2)
1329 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
1330     sa..contents = sb..contents & sa..size = sb..size"
1331     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1332     ensures "True" */
1333 {
1334     /*: assume "~(k1 ~= k2 | (EX v. (k1, v) : sa..contents))" */
1335     Object r1a = sa.put(k1, v1);
1336     boolean r2a = sa.containsKey(k2);
1337
1338     boolean r2b = sb.containsKey(k2);
1339     Object r1b = sb.put(k1, v1);
1340
1341     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1342         sb..size)" */
1343 }
1344 static void put_containsKey_between_s_43(AssociationList sa, AssociationList sb,
1345     Object k1, Object v1, Object k2)
1346 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
1347     sa..contents = sb..contents & sa..size = sb..size"
1348     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1349     ensures "True" */
1350 {
1351     Object r1a = sa.put(k1, v1);
1352     /*: 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)

```

```

1407  /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
1408      null &
1409          sa..contents = sb..contents & sa..size = sb..size"
1410      modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1411      ensures "True" */
1412  {
1413      /*: assume "k1 ~= k2 | (k1, v1) : sa..contents" */
1414      Object r1a = sa.put(k1, v1);
1415      Object r2a = sa.get(k2);
1416
1417      Object r2b = sb.get(k2);
1418      Object r1b = sb.put(k1, v1);
1419
1420      /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1421          sb..size" */
1422  }
1423
1424  static void put_get_pre_c_45(AssociationList sa, AssociationList sb, Object k1,
1425      Object v1, Object k2)
1426  /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
1427      null &
1428          sa..contents = sb..contents & sa..size = sb..size"
1429      modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1430      ensures "True" */
1431  {
1432      /*: assume "~(k1 ~= k2 | (k1, v1) : sa..contents)" */
1433      Object r1a = sa.put(k1, v1);
1434      Object r2a = sa.get(k2);
1435
1436      Object r2b = sb.get(k2);
1437      Object r1b = sb.put(k1, v1);
1438
1439      /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1440          sb..size)" */
1441  }
1442
1443  static void put_get_between_s_46(AssociationList sa, AssociationList sb, Object k1,
1444      Object v1, Object k2)
1445  /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
1446      null &
1447          sa..contents = sb..contents & sa..size = sb..size"
1448      modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1449      ensures "True" */
1450  {
1451      Object r1a = sa.put(k1, v1);
1452      /*: assume "k1 ~= k2 | r1a = v1" */
1453      Object r2a = sa.get(k2);
1454
1455      Object r2b = sb.get(k2);
1456      Object r1b = sb.put(k1, v1);
1457
1458      /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1459          sb..size" */
1460  }
1461
1462  static void put_get_between_c_46(AssociationList sa, AssociationList sb, Object k1,
1463      Object v1, Object k2)
1464  /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
1465      null &
1466          sa..contents = sb..contents & sa..size = sb..size"
1467      modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1468      ensures "True" */
1469  {
1470      Object r1a = sa.put(k1, v1);
1471      /*: 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,
1519 Object v1, Object k2, Object v2)
1519 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
1520 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 =
1532 sb..size)" */
1533 }
1534
1535 static void put_put_between_s_49(AssociationList sa, AssociationList sb, Object k1,
1536 Object v1, Object k2, Object v2)
1537 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
1538 null & v2 ~= null &
1539 sa..contents = sb..contents & sa..size = sb..size"
1540 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1541 ensures "True" */
1542 {
1543 Object r1a = sa.put(k1, v1);
1544 /*: assume "k1 ~= k2 | (r1a = v1 & v1 = v2)" */
1545 Object r2a = sa.put(k2, v2);
1546
1547 Object r2b = sb.put(k2, v2);
1548 Object r1b = sb.put(k1, v1);
1549
1550 /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1551 sb..size" */
1552 }
1553
1554 static void put_put_between_c_49(AssociationList sa, AssociationList sb, Object k1,
1555 Object v1, Object k2, Object v2)
1556 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
1557 null & v2 ~= null &
1558 sa..contents = sb..contents & sa..size = sb..size"
1559 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1560 ensures "True" */
1561 {
1562 Object r1a = sa.put(k1, v1);
1563 /*: assume "~(k1 ~= k2 | (r1a = v1 & v1 = v2))" */
1564 Object r2a = sa.put(k2, v2);
1565
1566 Object r2b = sb.put(k2, v2);
1567 Object r1b = sb.put(k1, v1);
1568
1569 /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1570 sb..size)" */
1571 }
1572
1573 static void put_put_post_s_50(AssociationList sa, AssociationList sb, Object k1,
1574 Object v1, Object k2, Object v2)
1575 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
1576 null & v2 ~= null &
1577 sa..contents = sb..contents & sa..size = sb..size"
1578 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1579 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,
1631     Object v1, Object k2, Object v2)
1632 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
1633     null & v2 ~= null &
1634     sa..contents = sb..contents & sa..size = sb..size"
1635     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1636     ensures "True" */
1637 {
1638     Object r1a = sa.put(k1, v1);
1639     /*: assume "k1 ~= k2 | (r1a = v1 & v1 = v2)" */
1640     sa.put(k2, v2);
1641
1642     sb.put(k2, v2);
1643     Object r1b = sb.put(k1, v1);
1644
1645     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1646 }
1647
1648 static void put_put_between_c_52(AssociationList sa, AssociationList sb, Object k1,
1649     Object v1, Object k2, Object v2)
1650 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
1651     null & v2 ~= null &
1652     sa..contents = sb..contents & sa..size = sb..size"
1653     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1654     ensures "True" */
1655 {
1656     Object r1a = sa.put(k1, v1);
1657     /*: assume "~(k1 ~= k2 | (r1a = v1 & v1 = v2))" */
1658     sa.put(k2, v2);
1659
1660     sb.put(k2, v2);
1661     Object r1b = sb.put(k1, v1);
1662
1663     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1664 }
1665
1666 static void put_put_post_s_53(AssociationList sa, AssociationList sb, Object k1,
1667     Object v1, Object k2, Object v2)
1668 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
1669     null & v2 ~= null &
1670     sa..contents = sb..contents & sa..size = sb..size"
1671     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1672     ensures "True" */
1673 {
1674     Object r1a = sa.put(k1, v1);
1675     sa.put(k2, v2);
1676     /*: assume "k1 ~= k2 | (r1a = v1 & v1 = v2)" */
1677
1678     sb.put(k2, v2);
1679     Object r1b = sb.put(k1, v1);
1680
1681     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1682 }
1683
1684 static void put_put_post_c_53(AssociationList sa, AssociationList sb, Object k1,
1685     Object v1, Object k2, Object v2)
1686 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
1687     null & v2 ~= null &
1688     sa..contents = sb..contents & sa..size = sb..size"
1689     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1690     ensures "True" */
1691 {
1692     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,
1695     Object v1, Object k2)
1696 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
1697     null &
1698     sa..contents = sb..contents & sa..size = sb..size"
1699     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1700     ensures "True" */
1701 {
1702     /*: assume "k1 ~= k2" */
1703     Object r1a = sa.put(k1, v1);
1704     Object r2a = sa.remove(k2);
1705
1706     Object r2b = sb.remove(k2);
1707     Object r1b = sb.put(k1, v1);
1708
1709     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1710     sb..size" */
1711 }
1712
1713 static void put_remove_pre_c_54(AssociationList sa, AssociationList sb, Object k1,
1714     Object v1, Object k2)
1715 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
1716     null &
1717     sa..contents = sb..contents & sa..size = sb..size"
1718     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1719     ensures "True" */
1720 {
1721     /*: assume "~(k1 ~= k2)" */
1722     Object r1a = sa.put(k1, v1);
1723     Object r2a = sa.remove(k2);
1724
1725     Object r2b = sb.remove(k2);
1726     Object r1b = sb.put(k1, v1);
1727
1728     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1729     sb..size)" */
1730 }
1731
1732 static void put_remove_between_s_55(AssociationList sa, AssociationList sb, Object
1733     k1, Object v1, Object k2)
1734 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
1735     null &
1736     sa..contents = sb..contents & sa..size = sb..size"
1737     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1738     ensures "True" */
1739 {
1740     Object r1a = sa.put(k1, v1);
1741     /*: assume "k1 ~= k2" */
1742     Object r2a = sa.remove(k2);
1743
1744     Object r2b = sb.remove(k2);
1745     Object r1b = sb.put(k1, v1);
1746
1747     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1748     sb..size" */
1749 }

```

```

1741 static void put_remove_between_c_55(AssociationList sa, AssociationList sb, Object
1742 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,
1807     Object v1, Object k2)
1808 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
1809     null &
1810     sa..contents = sb..contents & sa..size = sb..size"
1811     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1812     ensures "True" */
1813 {
1814     /*: assume "~(k1 ~= k2)" */
1815     Object r1a = sa.put(k1, v1);
1816     sa.remove(k2);
1817
1818     sb.remove(k2);
1819     Object r1b = sb.put(k1, v1);
1820
1821     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1822 }
1823
1824 static void put_remove_between_s_58(AssociationList sa, AssociationList sb, Object
1825     k1, Object v1, Object k2)
1826 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
1827     null &
1828     sa..contents = sb..contents & sa..size = sb..size"
1829     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1830     ensures "True" */
1831 {
1832     Object r1a = sa.put(k1, v1);
1833     /*: assume "k1 ~= k2" */
1834     sa.remove(k2);
1835
1836     sb.remove(k2);
1837     Object r1b = sb.put(k1, v1);
1838
1839     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1840 }
1841
1842 static void put_remove_between_c_58(AssociationList sa, AssociationList sb, Object
1843     k1, Object v1, Object k2)
1844 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
1845     null &
1846     sa..contents = sb..contents & sa..size = sb..size"
1847     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1848     ensures "True" */
1849 {
1850     Object r1a = sa.put(k1, v1);
1851     /*: assume "~(k1 ~= k2)" */
1852     sa.remove(k2);
1853
1854     sb.remove(k2);
1855     Object r1b = sb.put(k1, v1);
1856
1857     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1858 }

```

```

1854 static void put_remove_post_s_59(AssociationList sa, AssociationList sb, Object k1,
1855   Object v1, Object k2)
1856 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
1857   null &
1858     sa..contents = sb..contents & sa..size = sb..size"
1859   modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1860   ensures "True" */
1861 {
1862   Object r1a = sa.put(k1, v1);
1863   sa.remove(k2);
1864   /*: assume "k1 ~= k2" */
1865   sb.remove(k2);
1866   Object r1b = sb.put(k1, v1);
1867   /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1868 }
1869
1870 static void put_remove_post_c_59(AssociationList sa, AssociationList sb, Object k1,
1871   Object v1, Object k2)
1872 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
1873   null &
1874     sa..contents = sb..contents & sa..size = sb..size"
1875   modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1876   ensures "True" */
1877 {
1878   Object r1a = sa.put(k1, v1);
1879   sa.remove(k2);
1880   /*: assume "~(k1 ~= k2)" */
1881   sb.remove(k2);
1882   Object r1b = sb.put(k1, v1);
1883   /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1884 }
1885
1886 static void put_size_pre_s_60(AssociationList sa, AssociationList sb, Object k1,
1887   Object v1)
1888 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
1889   sa..contents = sb..contents & sa..size = sb..size"
1890   modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1891   ensures "True" */
1892 {
1893   /*: assume "EX v. (k1, v) : sa..contents" */
1894   Object r1a = sa.put(k1, v1);
1895   int r2a = sa.size();
1896   sb.remove(k1);
1897   int r2b = sb.size();
1898   Object r1b = sb.put(k1, v1);
1899   /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1900     sb..size" */
1901 }
1902
1903 static void put_size_pre_c_60(AssociationList sa, AssociationList sb, Object k1,
1904   Object v1)
1905 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
1906   sa..contents = sb..contents & sa..size = sb..size"
1907   modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1908   ensures "True" */
1909 {
1910   /*: assume "~(EX v. (k1, v) : sa..contents)" */
1911   Object r1a = sa.put(k1, v1);
1912   int r2a = sa.size();

```

```

1912     int r2b = sb.size();
1913     Object r1b = sb.put(k1, v1);
1914
1915     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
        sb..size)" */
1916 }
1917
1918 static void put_size_between_s_61(AssociationList sa, AssociationList sb, Object
    k1, Object v1)
1919 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
        sa..contents = sb..contents & sa..size = sb..size"
1920     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1921     ensures "True" */
1922 {
1923     Object r1a = sa.put(k1, v1);
1924     /*: assume "r1a ~= null" */
1925     int r2a = sa.size();
1926
1927     int r2b = sb.size();
1928     Object r1b = sb.put(k1, v1);
1929
1930     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
        sb..size" */
1931 }
1932
1933 static void put_size_between_c_61(AssociationList sa, AssociationList sb, Object
    k1, Object v1)
1934 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
        sa..contents = sb..contents & sa..size = sb..size"
1935     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1936     ensures "True" */
1937 {
1938     Object r1a = sa.put(k1, v1);
1939     /*: assume "~(r1a ~= null)" */
1940     int r2a = sa.size();
1941
1942     int r2b = sb.size();
1943     Object r1b = sb.put(k1, v1);
1944
1945     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
        sb..size)" */
1946 }
1947
1948 static void put_size_post_s_62(AssociationList sa, AssociationList sb, Object k1,
    Object v1)
1949 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
        sa..contents = sb..contents & sa..size = sb..size"
1950     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1951     ensures "True" */
1952 {
1953     Object r1a = sa.put(k1, v1);
1954     int r2a = sa.size();
1955     /*: assume "r1a ~= null" */
1956
1957     int r2b = sb.size();
1958     Object r1b = sb.put(k1, v1);
1959
1960     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
        sb..size" */
1961 }
1962
1963 static void put_size_post_c_62(AssociationList sa, AssociationList sb, Object k1,
    Object v1)
1964 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
        sa..contents = sb..contents & sa..size = sb..size"
1965
1966
1967
1968

```

```

1969     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1970     ensures "True" */
1971 {
1972     Object r1a = sa.put(k1, v1);
1973     int r2a = sa.size();
1974     /*: assume "~(r1a ~= null)" */
1975
1976     int r2b = sb.size();
1977     Object r1b = sb.put(k1, v1);
1978
1979     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1980         sb..size)" */
1981 }
1982
1983 static void put_containsKey_pre_s_63(AssociationList sa, AssociationList sb, Object
1984     k1, Object v1, Object k2)
1985 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
1986     sa..contents = sb..contents & sa..size = sb..size"
1987     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1988     ensures "True" */
1989 {
1990     /*: assume "k1 ~= k2 | (EX v. (k1, v) : sa..contents)" */
1991     sa.put(k1, v1);
1992     boolean r2a = sa.containsKey(k2);
1993
1994     boolean r2b = sb.containsKey(k2);
1995     sb.put(k1, v1);
1996
1997     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
1998 }
1999
2000 static void put_containsKey_pre_c_63(AssociationList sa, AssociationList sb, Object
2001     k1, Object v1, Object k2)
2002 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
2003     sa..contents = sb..contents & sa..size = sb..size"
2004     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2005     ensures "True" */
2006 {
2007     /*: assume "~(k1 ~= k2 | (EX v. (k1, v) : sa..contents))" */
2008     sa.put(k1, v1);
2009     boolean r2a = sa.containsKey(k2);
2010
2011     boolean r2b = sb.containsKey(k2);
2012     sb.put(k1, v1);
2013
2014     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2015 }
2016
2017 static void put_containsKey_between_s_64(AssociationList sa, AssociationList sb,
2018     Object k1, Object v1, Object k2)
2019 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
2020     sa..contents = sb..contents & sa..size = sb..size"
2021     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2022     ensures "True" */
2023 {
2024     sa.put(k1, v1);
2025     /*: assume "k1 ~= k2 | (EX v. (k1, v) : sa..(old contents))" */
2026     boolean r2a = sa.containsKey(k2);
2027
2028     boolean r2b = sb.containsKey(k2);
2029     sb.put(k1, v1);
2030
2031     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2032 }

```

```

2030 static void put_containsKey_between_c_64(AssociationList sa, AssociationList sb,
2031     Object k1, Object v1, Object k2)
2032 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
2033     sa..contents = sb..contents & sa..size = sb..size"
2034     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2035     ensures "True" */
2036 {
2037     sa.put(k1, v1);
2038     /*: assume "~(k1 ~= k2 | (EX v. (k1, v) : sa..(old contents)))" */
2039     boolean r2a = sa.containsKey(k2);
2040
2041     boolean r2b = sb.containsKey(k2);
2042     sb.put(k1, v1);
2043
2044     /*: assert "(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2045 }
2046
2047 static void put_containsKey_post_s_65(AssociationList sa, AssociationList sb,
2048     Object k1, Object v1, Object k2)
2049 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
2050     sa..contents = sb..contents & sa..size = sb..size"
2051     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2052     ensures "True" */
2053 {
2054     sa.put(k1, v1);
2055     boolean r2a = sa.containsKey(k2);
2056     /*: assume "k1 ~= k2 | (EX v. (k1, v) : sa..(old contents))" */
2057
2058     boolean r2b = sb.containsKey(k2);
2059     sb.put(k1, v1);
2060
2061     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2062 }
2063
2064 static void put_containsKey_post_c_65(AssociationList sa, AssociationList sb,
2065     Object k1, Object v1, Object k2)
2066 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null &
2067     sa..contents = sb..contents & sa..size = sb..size"
2068     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2069     ensures "True" */
2070 {
2071     sa.put(k1, v1);
2072     boolean r2a = sa.containsKey(k2);
2073     /*: assume "~(k1 ~= k2 | (EX v. (k1, v) : sa..(old contents)))" */
2074
2075     boolean r2b = sb.containsKey(k2);
2076     sb.put(k1, v1);
2077
2078     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2079 }
2080
2081 static void put_get_pre_s_66(AssociationList sa, AssociationList sb, Object k1,
2082     Object v1, Object k2)
2083 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
2084     null &
2085     sa..contents = sb..contents & sa..size = sb..size"
2086     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2087     ensures "True" */
2088 {
2089     /*: assume "k1 ~= k2 | (k1, v1) : sa..contents" */
2090     sa.put(k1, v1);
2091     Object r2a = sa.get(k2);
2092
2093     Object r2b = sb.get(k2);
2094     sb.put(k1, v1);

```

```

2090     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2091 }
2092
2093
2094 static void put_get_pre_c_66(AssociationList sa, AssociationList sb, Object k1,
2095     Object v1, Object k2)
2096 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
2097     null &
2098     sa..contents = sb..contents & sa..size = sb..size"
2099     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2100     ensures "True" */
2101 {
2102     /*: assume "~(k1 ~= k2 | (k1, v1) : sa..contents)" */
2103     sa.put(k1, v1);
2104     Object r2a = sa.get(k2);
2105
2106     Object r2b = sb.get(k2);
2107     sb.put(k1, v1);
2108
2109     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2110 }
2111
2112 static void put_get_between_s_67(AssociationList sa, AssociationList sb, Object k1,
2113     Object v1, Object k2)
2114 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
2115     null &
2116     sa..contents = sb..contents & sa..size = sb..size"
2117     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2118     ensures "True" */
2119 {
2120     sa.put(k1, v1);
2121     /*: assume "k1 ~= k2 | (k1, v1) : sa..(old contents)" */
2122     Object r2a = sa.get(k2);
2123
2124     Object r2b = sb.get(k2);
2125     sb.put(k1, v1);
2126
2127     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2128 }
2129
2130 static void put_get_between_c_67(AssociationList sa, AssociationList sb, Object k1,
2131     Object v1, Object k2)
2132 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
2133     null &
2134     sa..contents = sb..contents & sa..size = sb..size"
2135     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2136     ensures "True" */
2137 {
2138     sa.put(k1, v1);
2139     /*: assume "~(k1 ~= k2 | (k1, v1) : sa..(old contents))" */
2140     Object r2a = sa.get(k2);
2141
2142     Object r2b = sb.get(k2);
2143     sb.put(k1, v1);
2144
2145     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2146 }
2147
2148 static void put_get_post_s_68(AssociationList sa, AssociationList sb, Object k1,
2149     Object v1, Object k2)
2150 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
2151     null &
2152     sa..contents = sb..contents & sa..size = sb..size"
2153     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2154     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,
2159     Object v1, Object k2)
2160 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
2161     null &
2162     sa..contents = sb..contents & sa..size = sb..size"
2163     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2164     ensures "True" */
2165 {
2166     sa.put(k1, v1);
2167     Object r2a = sa.get(k2);
2168     /*: assume "~(k1 ~= k2 | (k1, v1) : sa..(old contents))" */
2169
2170     Object r2b = sb.get(k2);
2171     sb.put(k1, v1);
2172
2173     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2174 }
2175
2176 static void put_put_pre_s_69(AssociationList sa, AssociationList sb, Object k1,
2177     Object v1, Object k2, Object v2)
2178 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
2179     null & v2 ~= null &
2180     sa..contents = sb..contents & sa..size = sb..size"
2181     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2182     ensures "True" */
2183 {
2184     /*: assume "k1 ~= k2 | ((k1, v1) : sa..contents & v1 = v2)" */
2185     sa.put(k1, v1);
2186     Object r2a = sa.put(k2, v2);
2187
2188     Object r2b = sb.put(k2, v2);
2189     sb.put(k1, v1);
2190
2191     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2192 }
2193
2194 static void put_put_pre_c_69(AssociationList sa, AssociationList sb, Object k1,
2195     Object v1, Object k2, Object v2)
2196 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
2197     null & v2 ~= null &
2198     sa..contents = sb..contents & sa..size = sb..size"
2199     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2200     ensures "True" */
2201 {
2202     /*: assume "~(k1 ~= k2 | ((k1, v1) : sa..contents & v1 = v2))" */
2203     sa.put(k1, v1);
2204     Object r2a = sa.put(k2, v2);
2205
2206     Object r2b = sb.put(k2, v2);
2207     sb.put(k1, v1);
2208
2209     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2210 }

```

```

2206 static void put_put_between_s_70(AssociationList sa, AssociationList sb, Object k1,
2207     Object v1, Object k2, Object v2)
2208 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
2209     null & v2 ~= null &
2210     sa..contents = sb..contents & sa..size = sb..size"
2211     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2212     ensures "True" */
2213 {
2214     sa.put(k1, v1);
2215     /*: assume "k1 ~= k2 | ((k1, v1) : sa..(old contents) & v1 = v2)" */
2216     Object r2a = sa.put(k2, v2);
2217
2218     Object r2b = sb.put(k2, v2);
2219     sb.put(k1, v1);
2220
2221     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2222 }
2223
2224 static void put_put_between_c_70(AssociationList sa, AssociationList sb, Object k1,
2225     Object v1, Object k2, Object v2)
2226 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
2227     null & v2 ~= null &
2228     sa..contents = sb..contents & sa..size = sb..size"
2229     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2230     ensures "True" */
2231 {
2232     sa.put(k1, v1);
2233     /*: assume "~(k1 ~= k2 | ((k1, v1) : sa..(old contents) & v1 = v2))" */
2234     Object r2a = sa.put(k2, v2);
2235
2236     Object r2b = sb.put(k2, v2);
2237     sb.put(k1, v1);
2238
2239     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2240 }
2241
2242 static void put_put_post_s_71(AssociationList sa, AssociationList sb, Object k1,
2243     Object v1, Object k2, Object v2)
2244 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
2245     null & v2 ~= null &
2246     sa..contents = sb..contents & sa..size = sb..size"
2247     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2248     ensures "True" */
2249 {
2250     sa.put(k1, v1);
2251     Object r2a = sa.put(k2, v2);
2252     /*: assume "k1 ~= k2 | ((k1, v1) : sa..(old contents) & v1 = v2)" */
2253
2254     Object r2b = sb.put(k2, v2);
2255     sb.put(k1, v1);
2256
2257     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2258 }
2259
2260 static void put_put_post_c_71(AssociationList sa, AssociationList sb, Object k1,
2261     Object v1, Object k2, Object v2)
2262 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
2263     null & v2 ~= 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     sa.put(k1, v1);
2269     Object r2a = sa.put(k2, v2);
2270     /*: assume "~(k1 ~= k2 | ((k1, v1) : sa..(old contents) & v1 = v2))" */

```



```

2263     Object r2b = sb.put(k2, v2);
2264     sb.put(k1, v1);
2265
2266     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2267 }
2268
2269
2270 static void put_put_pre_s_72(AssociationList sa, AssociationList sb, Object k1,
2271     Object v1, Object k2, Object v2)
2272 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
2273     null & v2 ~= null &
2274     sa..contents = sb..contents & sa..size = sb..size"
2275     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2276     ensures "True" */
2277 {
2278     /*: assume "k1 ~= k2 | v1 = v2" */
2279     sa.put(k1, v1);
2280     sa.put(k2, v2);
2281
2282     sb.put(k2, v2);
2283     sb.put(k1, v1);
2284
2285     /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
2286 }
2287
2288 static void put_put_pre_c_72(AssociationList sa, AssociationList sb, Object k1,
2289     Object v1, Object k2, Object v2)
2290 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
2291     null & v2 ~= null &
2292     sa..contents = sb..contents & sa..size = sb..size"
2293     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2294     ensures "True" */
2295 {
2296     /*: assume "~(k1 ~= k2 | v1 = v2)" */
2297     sa.put(k1, v1);
2298     sa.put(k2, v2);
2299
2300     sb.put(k2, v2);
2301     sb.put(k1, v1);
2302
2303     /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
2304 }
2305
2306 static void put_put_between_s_73(AssociationList sa, AssociationList sb, Object k1,
2307     Object v1, Object k2, Object v2)
2308 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
2309     null & v2 ~= null &
2310     sa..contents = sb..contents & sa..size = sb..size"
2311     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2312     ensures "True" */
2313 {
2314     sa.put(k1, v1);
2315     /*: assume "k1 ~= k2 | v1 = v2" */
2316     sa.put(k2, v2);
2317
2318     sb.put(k2, v2);
2319     sb.put(k1, v1);
2320
2321     /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
2322 }
2323
2324 static void put_put_between_c_73(AssociationList sa, AssociationList sb, Object k1,
2325     Object v1, Object k2, Object v2)
2326 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
2327     null & v2 ~= null &

```

```

2320         sa..contents = sb..contents & sa..size = sb..size"
2321     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2322     ensures "True" */
2323 {
2324     sa.put(k1, v1);
2325     /*: assume "~(k1 ~= k2 | v1 = v2)" */
2326     sa.put(k2, v2);
2327
2328     sb.put(k2, v2);
2329     sb.put(k1, v1);
2330
2331     /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
2332 }
2333
2334 static void put_put_post_s_74(AssociationList sa, AssociationList sb, Object k1,
2335     Object v1, Object k2, Object v2)
2336 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
2337     null & v2 ~= null &
2338     sa..contents = sb..contents & sa..size = sb..size"
2339     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2340     ensures "True" */
2341 {
2342     sa.put(k1, v1);
2343     sa.put(k2, v2);
2344     /*: assume "k1 ~= k2 | v1 = v2" */
2345
2346     sb.put(k2, v2);
2347     sb.put(k1, v1);
2348
2349     /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
2350 }
2351
2352 static void put_put_post_c_74(AssociationList sa, AssociationList sb, Object k1,
2353     Object v1, Object k2, Object v2)
2354 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
2355     null & v2 ~= null &
2356     sa..contents = sb..contents & sa..size = sb..size"
2357     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2358     ensures "True" */
2359 {
2360     sa.put(k1, v1);
2361     sa.put(k2, v2);
2362     /*: assume "~(k1 ~= k2 | v1 = v2)" */
2363
2364     sb.put(k2, v2);
2365     sb.put(k1, v1);
2366
2367     /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
2368 }
2369
2370 static void put_remove_pre_s_75(AssociationList sa, AssociationList sb, Object k1,
2371     Object v1, Object k2)
2372 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
2373     null &
2374     sa..contents = sb..contents & sa..size = sb..size"
2375     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2376     ensures "True" */
2377 {
2378     /*: assume "k1 ~= k2" */
2379     sa.put(k1, v1);
2380     Object r2a = sa.remove(k2);
2381
2382     Object r2b = sb.remove(k2);
2383     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,
2383     Object v1, Object k2)
2384 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
2385     null &
2386     sa..contents = sb..contents & sa..size = sb..size"
2387     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2388     ensures "True" */
2389 {
2390     /*: assume "~(k1 ~= k2)" */
2391     sa.put(k1, v1);
2392     Object r2a = sa.remove(k2);
2393
2394     Object r2b = sb.remove(k2);
2395     sb.put(k1, v1);
2396
2397     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2398 }
2399
2400 static void put_remove_between_s_76(AssociationList sa, AssociationList sb, Object
2401     k1, Object v1, Object k2)
2402 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
2403     null &
2404     sa..contents = sb..contents & sa..size = sb..size"
2405     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2406     ensures "True" */
2407 {
2408     sa.put(k1, v1);
2409     /*: assume "k1 ~= k2" */
2410     Object r2a = sa.remove(k2);
2411
2412     Object r2b = sb.remove(k2);
2413     sb.put(k1, v1);
2414
2415     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2416 }
2417
2418 static void put_remove_between_c_76(AssociationList sa, AssociationList sb, Object
2419     k1, Object v1, Object k2)
2420 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
2421     null &
2422     sa..contents = sb..contents & sa..size = sb..size"
2423     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2424     ensures "True" */
2425 {
2426     sa.put(k1, v1);
2427     /*: assume "~(k1 ~= k2)" */
2428     Object r2a = sa.remove(k2);
2429
2430     Object r2b = sb.remove(k2);
2431     sb.put(k1, v1);
2432
2433     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2434 }
2435
2436 static void put_remove_post_s_77(AssociationList sa, AssociationList sb, Object k1,
2437     Object v1, Object k2)
2438 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
2439     null &
2440     sa..contents = sb..contents & sa..size = sb..size"
2441     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2442     ensures "True" */
2443 {

```

```

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,
2447     Object v1, Object k2)
2448 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
2449     null &
2450     sa..contents = sb..contents & sa..size = sb..size"
2451     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2452     ensures "True" */
2453 {
2454     sa.put(k1, v1);
2455     Object r2a = sa.remove(k2);
2456     /*: assume "~(k1 ~= k2)" */
2457
2458     Object r2b = sb.remove(k2);
2459     sb.put(k1, v1);
2460
2461     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2462 }
2463
2464 static void put_remove_pre_s_78(AssociationList sa, AssociationList sb, Object k1,
2465     Object v1, Object k2)
2466 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
2467     null &
2468     sa..contents = sb..contents & sa..size = sb..size"
2469     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2470     ensures "True" */
2471 {
2472     /*: assume "k1 ~= k2" */
2473     sa.put(k1, v1);
2474     sa.remove(k2);
2475
2476     sb.remove(k2);
2477     sb.put(k1, v1);
2478
2479     /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
2480 }
2481
2482 static void put_remove_pre_c_78(AssociationList sa, AssociationList sb, Object k1,
2483     Object v1, Object k2)
2484 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
2485     null &
2486     sa..contents = sb..contents & sa..size = sb..size"
2487     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2488     ensures "True" */
2489 {
2490     /*: assume "~(k1 ~= k2)" */
2491     sa.put(k1, v1);
2492     sa.remove(k2);
2493
2494     sb.remove(k2);
2495     sb.put(k1, v1);
2496
2497     /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
2498 }

```

```

2494 static void put_remove_between_s_79(AssociationList sa, AssociationList sb, Object
      k1, Object v1, Object k2)
2495 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
      null &
2496         sa..contents = sb..contents & sa..size = sb..size"
2497   modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2498   ensures "True" */
2499 {
2500   sa.put(k1, v1);
2501   /*: assume "k1 ~= k2" */
2502   sa.remove(k2);
2503
2504   sb.remove(k2);
2505   sb.put(k1, v1);
2506
2507   /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
2508 }
2509
2510 static void put_remove_between_c_79(AssociationList sa, AssociationList sb, Object
      k1, Object v1, Object k2)
2511 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
      null &
2512         sa..contents = sb..contents & sa..size = sb..size"
2513   modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2514   ensures "True" */
2515 {
2516   sa.put(k1, v1);
2517   /*: assume "~(k1 ~= k2)" */
2518   sa.remove(k2);
2519
2520   sb.remove(k2);
2521   sb.put(k1, v1);
2522
2523   /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
2524 }
2525
2526 static void put_remove_post_s_80(AssociationList sa, AssociationList sb, Object k1,
      Object v1, Object k2)
2527 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & v1 ~= null & k2 ~=
      null &
2528         sa..contents = sb..contents & sa..size = sb..size"
2529   modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2530   ensures "True" */
2531 {
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(AssociationList sa, AssociationList sb, Object k1,
      Object v1, Object k2)
2543 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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     sb.remove(k2);
2552     sb.put(k1, v1);
2553
2554     /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
2555 }
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 =
2684         sb..size)" */
2685 }
2686
2687 static void remove_containsKey_between_s_85(AssociationList sa, AssociationList sb,
2688     Object k1, Object k2)
2689 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
2690     sa..contents = sb..contents & sa..size = sb..size"
2691     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2692     ensures "True" */
2693 {
2694     Object r1a = sa.remove(k1);
2695     /*: assume "k1 ~= k2 | r1a = null" */
2696     boolean r2a = sa.containsKey(k2);
2697
2698     boolean r2b = sb.containsKey(k2);
2699     Object r1b = sb.remove(k1);
2700
2701     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2702         sb..size" */
2703 }
2704
2705 static void remove_containsKey_between_c_85(AssociationList sa, AssociationList sb,
2706     Object k1, Object k2)
2707 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
2708     sa..contents = sb..contents & sa..size = sb..size"
2709     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2710     ensures "True" */
2711 {
2712     Object r1a = sa.remove(k1);
2713     /*: assume "~(k1 ~= k2 | r1a = null)" */
2714     boolean r2a = sa.containsKey(k2);
2715
2716     boolean r2b = sb.containsKey(k2);
2717     Object r1b = sb.remove(k1);
2718
2719     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2720         sb..size)" */
2721 }
2722
2723 static void remove_containsKey_post_s_86(AssociationList sa, AssociationList sb,
2724     Object k1, Object k2)
2725 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
2726     sa..contents = sb..contents & sa..size = sb..size"
2727     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2728     ensures "True" */
2729 {
2730     Object r1a = sa.remove(k1);
2731     boolean r2a = sa.containsKey(k2);
2732     /*: assume "k1 ~= k2 | r1a = null" */
2733
2734     boolean r2b = sb.containsKey(k2);
2735     Object r1b = sb.remove(k1);
2736 }

```



```

2731     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2732         sb..size" */
2733 }
2734 static void remove_containsKey_post_c_86(AssociationList sa, AssociationList sb,
2735     Object k1, Object k2)
2736 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
2737     sa..contents = sb..contents & sa..size = sb..size"
2738     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2739     ensures "True" */
2740 {
2741     Object r1a = sa.remove(k1);
2742     boolean r2a = sa.containsKey(k2);
2743     /*: assume "~(k1 ~= k2 | r1a = null)" */
2744
2745     boolean r2b = sb.containsKey(k2);
2746     Object r1b = sb.remove(k1);
2747
2748     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2749         sb..size)" */
2750 }
2751 static void remove_get_pre_s_87(AssociationList sa, AssociationList sb, Object k1,
2752     Object k2)
2753 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
2754     sa..contents = sb..contents & sa..size = sb..size"
2755     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2756     ensures "True" */
2757 {
2758     /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
2759     Object r1a = sa.remove(k1);
2760     Object r2a = sa.get(k2);
2761
2762     Object r2b = sb.get(k2);
2763     Object r1b = sb.remove(k1);
2764
2765     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2766         sb..size" */
2767 }
2768 static void remove_get_pre_c_87(AssociationList sa, AssociationList sb, Object k1,
2769     Object k2)
2770 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
2771     sa..contents = sb..contents & sa..size = sb..size"
2772     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2773     ensures "True" */
2774 {
2775     /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
2776     Object r1a = sa.remove(k1);
2777     Object r2a = sa.get(k2);
2778
2779     Object r2b = sb.get(k2);
2780     Object r1b = sb.remove(k1);
2781
2782     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2783         sb..size)" */
2784 }
2785 static void remove_get_between_s_88(AssociationList sa, AssociationList sb, Object
2786     k1, Object k2)
2787 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
2788     sa..contents = sb..contents & sa..size = sb..size"
2789     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2790     ensures "True" */
2791 {

```

```

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 =
2796         sb..size" */
2797 }
2798
2799 static void remove_get_between_c_88(AssociationList sa, AssociationList sb, Object
2800     k1, Object k2)
2801 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
2802     sa..contents = sb..contents & sa..size = sb..size"
2803     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2804     ensures "True" */
2805 {
2806     Object r1a = sa.remove(k1);
2807     /*: assume "~(k1 ~= k2 | r1a = null)" */
2808     Object r2a = sa.get(k2);
2809
2810     Object r2b = sb.get(k2);
2811     Object r1b = sb.remove(k1);
2812
2813     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2814         sb..size)" */
2815 }
2816
2817 static void remove_get_post_s_89(AssociationList sa, AssociationList sb, Object k1,
2818     Object k2)
2819 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
2820     sa..contents = sb..contents & sa..size = sb..size"
2821     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2822     ensures "True" */
2823 {
2824     Object r1a = sa.remove(k1);
2825     Object r2a = sa.get(k2);
2826     /*: assume "k1 ~= k2 | r1a = null" */
2827
2828     Object r2b = sb.get(k2);
2829     Object r1b = sb.remove(k1);
2830
2831     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2832         sb..size" */
2833 }
2834
2835 static void remove_get_post_c_89(AssociationList sa, AssociationList sb, Object k1,
2836     Object k2)
2837 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
2838     sa..contents = sb..contents & sa..size = sb..size"
2839     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2840     ensures "True" */
2841 {
2842     Object r1a = sa.remove(k1);
2843     Object r2a = sa.get(k2);
2844     /*: assume "~(k1 ~= k2 | r1a = null)" */
2845
2846     Object r2b = sb.get(k2);
2847     Object r1b = sb.remove(k1);
2848
2849     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2850         sb..size)" */
2851 }

```

```

2846 static void remove_put_pre_s_90(AssociationList sa, AssociationList sb, Object k1,
2847 Object k2, Object v2)
2848 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
2849 null &
2850         sa..contents = sb..contents & sa..size = sb..size"
2851 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2852 ensures "True" */
2853 {
2854     /*: assume "k1 ~= k2" */
2855     Object r1a = sa.remove(k1);
2856     Object r2a = sa.put(k2, v2);
2857
2858     Object r2b = sb.put(k2, v2);
2859     Object r1b = sb.remove(k1);
2860
2861     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2862 sb..size" */
2863 }
2864
2865 static void remove_put_pre_c_90(AssociationList sa, AssociationList sb, Object k1,
2866 Object k2, Object v2)
2867 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
2868 null &
2869         sa..contents = sb..contents & sa..size = sb..size"
2870 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2871 ensures "True" */
2872 {
2873     /*: assume "~(k1 ~= k2)" */
2874     Object r1a = sa.remove(k1);
2875     Object r2a = sa.put(k2, v2);
2876
2877     Object r2b = sb.put(k2, v2);
2878     Object r1b = sb.remove(k1);
2879
2880     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2881 sb..size)" */
2882 }
2883
2884 static void remove_put_between_s_91(AssociationList sa, AssociationList sb, Object
2885 k1, Object k2, Object v2)
2886 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
2887 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     Object r1a = sa.remove(k1);
2893     /*: assume "k1 ~= k2" */
2894     Object r2a = sa.put(k2, v2);
2895
2896     Object r2b = sb.put(k2, v2);
2897     Object r1b = sb.remove(k1);
2898
2899     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2900 sb..size" */
2901 }
2902
2903 static void remove_put_between_c_91(AssociationList sa, AssociationList sb, Object
2904 k1, Object k2, Object v2)
2905 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
2906 null &
2907         sa..contents = sb..contents & sa..size = sb..size"
2908 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2909 ensures "True" */
2910 {

```

```

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 =
2908         sb..size)" */
2909 }
2910
2911 static void remove_put_post_s_92(AssociationList sa, AssociationList sb, Object k1,
2912     Object k2, Object v2)
2913 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
2914     null &
2915         sa..contents = sb..contents & sa..size = sb..size"
2916     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2917     ensures "True" */
2918 {
2919     Object r1a = sa.remove(k1);
2920     Object r2a = sa.put(k2, v2);
2921     /*: assume "k1 ~= k2" */
2922
2923     Object r2b = sb.put(k2, v2);
2924     Object r1b = sb.remove(k1);
2925
2926     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2927         sb..size" */
2928 }
2929
2930 static void remove_put_post_c_92(AssociationList sa, AssociationList sb, Object k1,
2931     Object k2, Object v2)
2932 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
2933     null &
2934         sa..contents = sb..contents & sa..size = sb..size"
2935     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2936     ensures "True" */
2937 {
2938     Object r1a = sa.remove(k1);
2939     Object r2a = sa.put(k2, v2);
2940     /*: assume "~(k1 ~= k2)" */
2941
2942     Object r2b = sb.put(k2, v2);
2943     Object r1b = sb.remove(k1);
2944
2945     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2946         sb..size)" */
2947 }
2948
2949 static void remove_put_pre_s_93(AssociationList sa, AssociationList sb, Object k1,
2950     Object k2, Object v2)
2951 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
2952     null &
2953         sa..contents = sb..contents & sa..size = sb..size"
2954     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2955     ensures "True" */
2956 {
2957     /*: assume "k1 ~= k2" */
2958     Object r1a = sa.remove(k1);
2959     sa.put(k2, v2);
2960
2961     sb.put(k2, v2);
2962     Object r1b = sb.remove(k1);
2963
2964     /*: 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,
2959 Object k2, Object v2)
2959 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
2960 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
2975 k1, Object k2, Object v2)
2975 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
2976 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
2991 k1, Object k2, Object v2)
2991 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
2992 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,
3007 Object k2, Object v2)
3007 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
3008 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,
3023     Object k2, Object v2)
3024 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
3025     null &
3026     sa..contents = sb..contents & sa..size = sb..size"
3027     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3028     ensures "True" */
3029 {
3030     Object r1a = sa.remove(k1);
3031     sa.put(k2, v2);
3032     /*: assume "~(k1 ~= k2)" */
3033
3034     sb.put(k2, v2);
3035     Object r1b = sb.remove(k1);
3036
3037     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
3038 }
3039
3040 static void remove_remove_pre_s_96(AssociationList sa, AssociationList sb, Object
3041     k1, Object k2)
3042 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3043     sa..contents = sb..contents & sa..size = sb..size"
3044     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3045     ensures "True" */
3046 {
3047     /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
3048     Object r1a = sa.remove(k1);
3049     Object r2a = sa.remove(k2);
3050
3051     Object r2b = sb.remove(k2);
3052     Object r1b = sb.remove(k1);
3053
3054     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
3055     sb..size" */
3056 }
3057
3058 static void remove_remove_pre_c_96(AssociationList sa, AssociationList sb, Object
3059     k1, Object k2)
3060 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3061     sa..contents = sb..contents & sa..size = sb..size"
3062     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3063     ensures "True" */
3064 {
3065     /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
3066     Object r1a = sa.remove(k1);
3067     Object r2a = sa.remove(k2);
3068
3069     Object r2b = sb.remove(k2);
3070     Object r1b = sb.remove(k1);
3071
3072     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
3073     sb..size)" */
3074 }
3075
3076 static void remove_remove_between_s_97(AssociationList sa, AssociationList sb,
3077     Object k1, Object k2)

```

```

3071  /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3072          sa..contents = sb..contents & sa..size = sb..size"
3073  modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3074  ensures "True" */
3075  {
3076      Object r1a = sa.remove(k1);
3077      /*: assume "k1 ~= k2 | r1a = null" */
3078      Object r2a = sa.remove(k2);
3079
3080      Object r2b = sb.remove(k2);
3081      Object r1b = sb.remove(k1);
3082
3083      /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
          sb..size" */
3084  }
3085
3086  static void remove_remove_between_c_97(AssociationList sa, AssociationList sb,
          Object k1, Object k2)
3087  /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
          sa..contents = sb..contents & sa..size = sb..size"
3088  modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3089  ensures "True" */
3090  {
3091      Object r1a = sa.remove(k1);
3092      /*: assume "~(k1 ~= k2 | r1a = null)" */
3093      Object r2a = sa.remove(k2);
3094
3095      Object r2b = sb.remove(k2);
3096      Object r1b = sb.remove(k1);
3097
3098      /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
          sb..size)" */
3099  }
3100
3101
3102  static void remove_remove_post_s_98(AssociationList sa, AssociationList sb, Object
          k1, Object k2)
3103  /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
          sa..contents = sb..contents & sa..size = sb..size"
3104  modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3105  ensures "True" */
3106  {
3107      Object r1a = sa.remove(k1);
3108      Object r2a = sa.remove(k2);
3109      /*: assume "k1 ~= k2 | r1a = null" */
3110
3111      Object r2b = sb.remove(k2);
3112      Object r1b = sb.remove(k1);
3113
3114      /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
          sb..size" */
3115  }
3116
3117
3118  static void remove_remove_post_c_98(AssociationList sa, AssociationList sb, Object
          k1, Object k2)
3119  /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
          sa..contents = sb..contents & sa..size = sb..size"
3120  modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3121  ensures "True" */
3122  {
3123      Object r1a = sa.remove(k1);
3124      Object r2a = sa.remove(k2);
3125      /*: assume "~(k1 ~= k2 | r1a = null)" */
3126
3127      Object r2b = sb.remove(k2);
3128      Object r1b = sb.remove(k1);
3129

```

```

3130
3131     /*: assert "~(ria = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
3132         sb..size)" */
3133 }
3134 static void remove_remove_pre_s_99(AssociationList sa, AssociationList sb, Object
3135     k1, Object k2)
3136 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3137     sa..contents = sb..contents & sa..size = sb..size"
3138     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3139     ensures "True" */
3140 {
3141     /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
3142     Object r1a = sa.remove(k1);
3143     sa.remove(k2);
3144
3145     sb.remove(k2);
3146     Object r1b = sb.remove(k1);
3147
3148     /*: assert "ria = r1b & sa..contents = sb..contents & sa..size = sb..size" */
3149 }
3150 static void remove_remove_pre_c_99(AssociationList sa, AssociationList sb, Object
3151     k1, Object k2)
3152 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3153     sa..contents = sb..contents & sa..size = sb..size"
3154     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3155     ensures "True" */
3156 {
3157     /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
3158     Object r1a = sa.remove(k1);
3159     sa.remove(k2);
3160
3161     sb.remove(k2);
3162     Object r1b = sb.remove(k1);
3163
3164     /*: assert "~(ria = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
3165 }
3166 static void remove_remove_between_s_100(AssociationList sa, AssociationList sb,
3167     Object k1, Object k2)
3168 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3169     sa..contents = sb..contents & sa..size = sb..size"
3170     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3171     ensures "True" */
3172 {
3173     Object r1a = sa.remove(k1);
3174     /*: assume "k1 ~= k2 | r1a = null" */
3175     sa.remove(k2);
3176
3177     sb.remove(k2);
3178     Object r1b = sb.remove(k1);
3179
3180     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
3181 }
3182 static void remove_remove_between_c_100(AssociationList sa, AssociationList sb,
3183     Object k1, Object k2)
3184 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3185     sa..contents = sb..contents & sa..size = sb..size"
3186     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3187     ensures "True" */
3188 {
3189     Object r1a = sa.remove(k1);
3190     /*: 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(AssociationList sa, AssociationList sb, Object
3199     k1, Object k2)
3200 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3201     sa..contents = sb..contents & sa..size = sb..size"
3202     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3203     ensures "True" */
3204 {
3205     Object r1a = sa.remove(k1);
3206     sa.remove(k2);
3207     /*: assume "k1 ~= k2 | r1a = null" */
3208
3209     sb.remove(k2);
3210     Object r1b = sb.remove(k1);
3211
3212     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
3213 }
3214
3215 static void remove_remove_post_c_101(AssociationList sa, AssociationList sb, Object
3216     k1, Object k2)
3217 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3218     sa..contents = sb..contents & sa..size = sb..size"
3219     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3220     ensures "True" */
3221 {
3222     Object r1a = sa.remove(k1);
3223     sa.remove(k2);
3224     /*: assume "~(k1 ~= k2 | r1a = null)" */
3225
3226     sb.remove(k2);
3227     Object r1b = sb.remove(k1);
3228
3229     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
3230 }
3231
3232 static void remove_size_pre_s_102(AssociationList sa, AssociationList sb, Object k1)
3233 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
3234     sa..contents = sb..contents & sa..size = sb..size"
3235     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3236     ensures "True" */
3237 {
3238     /*: assume "~(EX v. (k1, v) : sa..contents)" */
3239     Object r1a = sa.remove(k1);
3240     int r2a = sa.size();
3241
3242     int r2b = sb.size();
3243     Object r1b = sb.remove(k1);
3244
3245     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
3246     sb..size" */
3247 }
3248
3249 static void remove_size_pre_c_102(AssociationList sa, AssociationList sb, Object k1)
3250 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
3251     sa..contents = sb..contents & sa..size = sb..size"
3252     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3253     ensures "True" */
3254 {

```

```

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 =
3260         sb..size)" */
3261 }
3262
3263 static void remove_size_between_s_103(AssociationList sa, AssociationList sb,
3264     Object k1)
3265 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
3266     sa..contents = sb..contents & sa..size = sb..size"
3267     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3268     ensures "True" */
3269 {
3270     Object r1a = sa.remove(k1);
3271     /*: assume "r1a = null" */
3272     int r2a = sa.size();
3273
3274     int r2b = sb.size();
3275     Object r1b = sb.remove(k1);
3276
3277     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
3278         sb..size" */
3279 }
3280
3281 static void remove_size_between_c_103(AssociationList sa, AssociationList sb,
3282     Object k1)
3283 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= 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     Object r1a = sa.remove(k1);
3289     /*: assume "~(r1a = null)" */
3290     int r2a = sa.size();
3291
3292     int r2b = sb.size();
3293     Object r1b = sb.remove(k1);
3294
3295     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
3296         sb..size)" */
3297 }
3298
3299 static void remove_size_post_s_104(AssociationList sa, AssociationList sb, Object
3300     k1)
3301 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
3302     sa..contents = sb..contents & sa..size = sb..size"
3303     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3304     ensures "True" */
3305 {
3306     Object r1a = sa.remove(k1);
3307     int r2a = sa.size();
3308     /*: assume "r1a = null" */
3309
3310     int r2b = sb.size();
3311     Object r1b = sb.remove(k1);
3312
3313     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
3314         sb..size" */
3315 }

```

```

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     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3371 }
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     Object r2b = sb.get(k2);
3432     sb.remove(k1);
3433
3434     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3435 }
3436
3437
3438 static void remove_get_pre_c_108(AssociationList sa, AssociationList sb, Object k1,
3439     Object k2)
3440 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3441     sa..contents = sb..contents & sa..size = sb..size"
3442     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3443     ensures "True" */
3444 {
3445     /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
3446     sa.remove(k1);
3447     Object r2a = sa.get(k2);
3448
3449     Object r2b = sb.get(k2);
3450     sb.remove(k1);
3451
3452     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3453 }
3454
3455 static void remove_get_between_s_109(AssociationList sa, AssociationList sb, Object
3456     k1, Object k2)
3457 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3458     sa..contents = sb..contents & sa..size = sb..size"
3459     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3460     ensures "True" */
3461 {
3462     sa.remove(k1);
3463     /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents))" */
3464     Object r2a = sa.get(k2);
3465
3466     Object r2b = sb.get(k2);
3467     sb.remove(k1);
3468
3469     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3470 }
3471
3472 static void remove_get_between_c_109(AssociationList sa, AssociationList sb, Object
3473     k1, Object k2)
3474 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3475     sa..contents = sb..contents & sa..size = sb..size"
3476     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3477     ensures "True" */
3478 {
3479     sa.remove(k1);
3480     /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents)))" */
3481     Object r2a = sa.get(k2);
3482
3483     Object r2b = sb.get(k2);
3484     sb.remove(k1);
3485
3486     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3487 }
3488
3489 static void remove_get_post_s_110(AssociationList sa, AssociationList sb, Object
3490     k1, Object k2)
3491 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3492     sa..contents = sb..contents & sa..size = sb..size"
3493     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3494     ensures "True" */
3495 {

```

```

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
3503     k1, Object k2)
3504 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3505     sa..contents = sb..contents & sa..size = sb..size"
3506     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3507     ensures "True" */
3508 {
3509     sa.remove(k1);
3510     Object r2a = sa.get(k2);
3511     /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents)))" */
3512
3513     Object r2b = sb.get(k2);
3514     sb.remove(k1);
3515
3516     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3517 }
3518
3519 static void remove_put_pre_s_111(AssociationList sa, AssociationList sb, Object k1,
3520     Object k2, Object v2)
3521 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
3522     null &
3523     sa..contents = sb..contents & sa..size = sb..size"
3524     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3525     ensures "True" */
3526 {
3527     /*: assume "k1 ~= k2" */
3528     sa.remove(k1);
3529     Object r2a = sa.put(k2, v2);
3530
3531     Object r2b = sb.put(k2, v2);
3532     sb.remove(k1);
3533
3534     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3535 }
3536
3537 static void remove_put_pre_c_111(AssociationList sa, AssociationList sb, Object k1,
3538     Object k2, Object v2)
3539 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
3540     null &
3541     sa..contents = sb..contents & sa..size = sb..size"
3542     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3543     ensures "True" */
3544 {
3545     /*: assume "~(k1 ~= k2)" */
3546     sa.remove(k1);
3547     Object r2a = sa.put(k2, v2);
3548
3549     Object r2b = sb.put(k2, v2);
3550     sb.remove(k1);
3551
3552     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3553 }
3554
3555 static void remove_put_between_s_112(AssociationList sa, AssociationList sb, Object
3556     k1, Object k2, Object v2)

```

```

3551  /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(AssociationList sa, AssociationList sb, Object
      k1, Object k2, Object v2)
3567  /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(AssociationList sa, AssociationList sb, Object
      k1, Object k2, Object v2)
3583  /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(AssociationList sa, AssociationList sb, Object
      k1, Object k2, Object v2)
3599  /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(AssociationList sa, AssociationList sb, Object k1,
3615     Object k2, Object v2)
3616 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
3617     null &
3618     sa..contents = sb..contents & sa..size = sb..size"
3619     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3620     ensures "True" */
3621 {
3622     /*: assume "k1 ~= k2" */
3623     sa.remove(k1);
3624     sa.put(k2, v2);
3625
3626     sb.put(k2, v2);
3627     sb.remove(k1);
3628
3629     /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
3630 }
3631
3632 static void remove_put_pre_c_114(AssociationList sa, AssociationList sb, Object k1,
3633     Object k2, Object v2)
3634 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
3635     null &
3636     sa..contents = sb..contents & sa..size = sb..size"
3637     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3638     ensures "True" */
3639 {
3640     /*: assume "~(k1 ~= k2)" */
3641     sa.remove(k1);
3642     sa.put(k2, v2);
3643
3644     sb.put(k2, v2);
3645     sb.remove(k1);
3646
3647     /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
3648 }
3649
3650 static void remove_put_between_s_115(AssociationList sa, AssociationList sb, Object
3651     k1, Object k2, Object v2)
3652 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
3653     null &
3654     sa..contents = sb..contents & sa..size = sb..size"
3655     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3656     ensures "True" */
3657 {
3658     sa.remove(k1);
3659     /*: assume "k1 ~= k2" */
3660     sa.put(k2, v2);
3661
3662     sb.put(k2, v2);
3663     sb.remove(k1);
3664
3665     /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
3666 }
3667
3668 static void remove_put_between_c_115(AssociationList sa, AssociationList sb, Object
3669     k1, Object k2, Object v2)
3670 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
3671     null &
3672     sa..contents = sb..contents & sa..size = sb..size"
3673     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
3679     k1, Object k2, Object v2)
3680 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
3681     null &
3682     sa..contents = sb..contents & sa..size = sb..size"
3683     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3684     ensures "True" */
3685 {
3686     sa.remove(k1);
3687     sa.put(k2, v2);
3688     /*: assume "k1 ~= k2" */
3689
3690     sb.put(k2, v2);
3691     sb.remove(k1);
3692
3693     /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
3694 }
3695
3696 static void remove_put_post_c_116(AssociationList sa, AssociationList sb, Object
3697     k1, Object k2, Object v2)
3698 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null & v2 ~=
3699     null &
3700     sa..contents = sb..contents & sa..size = sb..size"
3701     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3702     ensures "True" */
3703 {
3704     sa.remove(k1);
3705     sa.put(k2, v2);
3706     /*: assume "~(k1 ~= k2)" */
3707
3708     sb.put(k2, v2);
3709     sb.remove(k1);
3710
3711     /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
3712 }
3713
3714 static void remove_remove_pre_s_117(AssociationList sa, AssociationList sb, Object
3715     k1, Object k2)
3716 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3717     sa..contents = sb..contents & sa..size = sb..size"
3718     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3719     ensures "True" */
3720 {
3721     /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
3722     sa.remove(k1);
3723     Object r2a = sa.remove(k2);
3724
3725     Object r2b = sb.remove(k2);
3726     sb.remove(k1);
3727
3728     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3729 }
3730

```

```

3726 static void remove_remove_pre_c_117(AssociationList sa, AssociationList sb, Object
3727 k1, Object k2)
3728 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3729 sa..contents = sb..contents & sa..size = sb..size"
3730 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3731 ensures "True" */
3732 {
3733 /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
3734 sa.remove(k1);
3735 Object r2a = sa.remove(k2);
3736
3737 Object r2b = sb.remove(k2);
3738 sb.remove(k1);
3739
3740 /*: assert "(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3741 }
3742
3743 static void remove_remove_between_s_118(AssociationList sa, AssociationList sb,
3744 Object k1, Object k2)
3745 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3746 sa..contents = sb..contents & sa..size = sb..size"
3747 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3748 ensures "True" */
3749 {
3750 sa.remove(k1);
3751 /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents))" */
3752 Object r2a = sa.remove(k2);
3753
3754 Object r2b = sb.remove(k2);
3755 sb.remove(k1);
3756
3757 /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3758 }
3759
3760 static void remove_remove_between_c_118(AssociationList sa, AssociationList sb,
3761 Object k1, Object k2)
3762 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3763 sa..contents = sb..contents & sa..size = sb..size"
3764 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3765 ensures "True" */
3766 {
3767 sa.remove(k1);
3768 /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents)))" */
3769 Object r2a = sa.remove(k2);
3770
3771 Object r2b = sb.remove(k2);
3772 sb.remove(k1);
3773
3774 /*: assert "(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3775 }
3776
3777 static void remove_remove_post_s_119(AssociationList sa, AssociationList sb, Object
3778 k1, Object k2)
3779 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3780 sa..contents = sb..contents & sa..size = sb..size"
3781 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3782 ensures "True" */
3783 {
3784 sa.remove(k1);
3785 Object r2a = sa.remove(k2);
3786 /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents))" */
3787
3788 Object r2b = sb.remove(k2);
3789 sb.remove(k1);
3790

```

```

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
3791     k1, Object k2)
3792 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3793     sa..contents = sb..contents & sa..size = sb..size"
3794     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3795     ensures "True" */
3796 {
3797     sa.remove(k1);
3798     Object r2a = sa.remove(k2);
3799     /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents)))" */
3800
3801     Object r2b = sb.remove(k2);
3802     sb.remove(k1);
3803
3804     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3805 }
3806
3807 static void remove_remove_pre_s_120(AssociationList sa, AssociationList sb, Object
3808     k1, Object k2)
3809 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3810     sa..contents = sb..contents & sa..size = sb..size"
3811     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3812     ensures "True" */
3813 {
3814     /*: assume "True" */
3815     sa.remove(k1);
3816     sa.remove(k2);
3817
3818     sb.remove(k2);
3819     sb.remove(k1);
3820
3821     /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
3822 }
3823
3824 static void remove_remove_pre_c_120(AssociationList sa, AssociationList sb, Object
3825     k1, Object k2)
3826 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3827     sa..contents = sb..contents & sa..size = sb..size"
3828     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3829     ensures "True" */
3830 {
3831     /*: assume "~(True)" */
3832     sa.remove(k1);
3833     sa.remove(k2);
3834
3835     sb.remove(k2);
3836     sb.remove(k1);
3837
3838     /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
3839 }
3840
3841 static void remove_remove_between_s_121(AssociationList sa, AssociationList sb,
3842     Object k1, Object k2)
3843 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null & k2 ~= null &
3844     sa..contents = sb..contents & sa..size = sb..size"
3845     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3846     ensures "True" */
3847 {
3848     sa.remove(k1);
3849     /*: assume "True" */
3850     sa.remove(k2);

```

```

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(AssociationList sa, AssociationList sb,
    Object k1, Object k2)
3855 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(AssociationList sa, AssociationList sb, Object
    k1, Object k2)
3871 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(AssociationList sa, AssociationList sb, Object
    k1, Object k2)
3887 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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(AssociationList sa, AssociationList sb, Object k1)
3903 /*: requires "sa ~= null & sb ~= null & sa ~= sb & 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
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,
3935     Object k1)
3936 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
3937     sa..contents = sb..contents & sa..size = sb..size"
3938     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3939     ensures "True" */
3940 {
3941     sa.remove(k1);
3942     /*: assume "~(EX v. (k1, v) : sa..(old contents))" */
3943     int r2a = sa.size();
3944
3945     int r2b = sb.size();
3946     sb.remove(k1);
3947
3948     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3949 }
3950
3951 static void remove_size_between_c_124(AssociationList sa, AssociationList sb,
3952     Object k1)
3953 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
3954     sa..contents = sb..contents & sa..size = sb..size"
3955     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3956     ensures "True" */
3957 {
3958     sa.remove(k1);
3959     /*: assume "~(EX v. (k1, v) : sa..(old contents)))" */
3960     int r2a = sa.size();
3961
3962     int r2b = sb.size();
3963     sb.remove(k1);
3964
3965     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3966 }
3967
3968 static void remove_size_post_s_125(AssociationList sa, AssociationList sb, Object
3969     k1)
3970 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
3971     sa..contents = sb..contents & sa..size = sb..size"
3972     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3973     ensures "True" */
3974 {

```

```

3972     sa.remove(k1);
3973     int r2a = sa.size();
3974     /*: assume "~(EX v. (k1, v) : sa..(old contents))" */
3975
3976     int r2b = sb.size();
3977     sb.remove(k1);
3978
3979     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3980 }
3981
3982 static void remove_size_post_c_125(AssociationList sa, AssociationList sb, Object
    k1)
3983 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k1 ~= null &
3984     sa..contents = sb..contents & sa..size = sb..size"
3985     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3986     ensures "True" */
3987 {
3988     sa.remove(k1);
3989     int r2a = sa.size();
3990     /*: assume "~(~(EX v. (k1, v) : sa..(old contents)))" */
3991
3992     int r2b = sb.size();
3993     sb.remove(k1);
3994
3995     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3996 }
3997
3998 static void size_containsKey_pre_s_126(AssociationList sa, AssociationList sb,
    Object k2)
3999 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
4000     sa..contents = sb..contents & sa..size = sb..size"
4001     ensures "True" */
4002 {
4003     /*: assume "True" */
4004     int r1a = sa.size();
4005     boolean r2a = sa.containsKey(k2);
4006
4007     boolean r2b = sb.containsKey(k2);
4008     int r1b = sb.size();
4009
4010     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
4011     sb..size" */
4012 }
4013
4014 static void size_containsKey_pre_c_126(AssociationList sa, AssociationList sb,
    Object k2)
4015 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
4016     sa..contents = sb..contents & sa..size = sb..size"
4017     ensures "True" */
4018 {
4019     /*: assume "~(True)" */
4020     int r1a = sa.size();
4021     boolean r2a = sa.containsKey(k2);
4022
4023     boolean r2b = sb.containsKey(k2);
4024     int r1b = sb.size();
4025
4026     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
4027     sb..size)" */
4028 }
4029
4030 static void size_containsKey_between_s_127(AssociationList sa, AssociationList sb,
    Object k2)
4031 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
4032     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 &
         sa..contents = sb..contents & sa..size = sb..size"
4045     ensures "True" */
4046 {
4047     int r1a = sa.size();
4048     /*: assume "~(True)" */
4049     boolean r2a = sa.containsKey(k2);
4050
4051     boolean r2b = sb.containsKey(k2);
4052     int r1b = sb.size();
4053
4054     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
         sb..size)" */
4055 }
4056
4057 static void size_containsKey_post_s_128(AssociationList sa, AssociationList sb,
         Object k2)
4058 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
         sa..contents = sb..contents & sa..size = sb..size"
4059     ensures "True" */
4060 {
4061     int r1a = sa.size();
4062     boolean r2a = sa.containsKey(k2);
4063     /*: assume "True" */
4064
4065     boolean r2b = sb.containsKey(k2);
4066     int r1b = sb.size();
4067
4068     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
         sb..size" */
4069 }
4070
4071 static void size_containsKey_post_c_128(AssociationList sa, AssociationList sb,
         Object k2)
4072 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
         sa..contents = sb..contents & sa..size = sb..size"
4073     ensures "True" */
4074 {
4075     int r1a = sa.size();
4076     boolean r2a = sa.containsKey(k2);
4077     /*: assume "~(True)" */
4078
4079     boolean r2b = sb.containsKey(k2);
4080     int r1b = sb.size();
4081
4082     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
         sb..size)" */
4083 }
4084
4085 static void size_get_pre_s_129(AssociationList sa, AssociationList sb, Object k2)

```

```

4089  /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
4090          sa..contents = sb..contents & sa..size = sb..size"
4091  ensures "True" */
4092  {
4093    /*: assume "True" */
4094    int r1a = sa.size();
4095    Object r2a = sa.get(k2);
4096
4097    Object r2b = sb.get(k2);
4098    int r1b = sb.size();
4099
4100    /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
          sb..size" */
4101  }
4102
4103  static void size_get_pre_c_129(AssociationList sa, AssociationList sb, Object k2)
4104  /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
          sa..contents = sb..contents & sa..size = sb..size"
4105  ensures "True" */
4106  {
4107    /*: assume "~(True)" */
4108    int r1a = sa.size();
4109    Object r2a = sa.get(k2);
4110
4111    Object r2b = sb.get(k2);
4112    int r1b = sb.size();
4113
4114    /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
          sb..size)" */
4115  }
4116
4117  static void size_get_between_s_130(AssociationList sa, AssociationList sb, Object
          k2)
4118  /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
          sa..contents = sb..contents & sa..size = sb..size"
4119  ensures "True" */
4120  {
4121    int r1a = sa.size();
4122    /*: assume "True" */
4123    Object r2a = sa.get(k2);
4124
4125    Object r2b = sb.get(k2);
4126    int r1b = sb.size();
4127
4128    /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
          sb..size" */
4129  }
4130
4131  static void size_get_between_c_130(AssociationList sa, AssociationList sb, Object
          k2)
4132  /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
          sa..contents = sb..contents & sa..size = sb..size"
4133  ensures "True" */
4134  {
4135    int r1a = sa.size();
4136    /*: assume "~(True)" */
4137    Object r2a = sa.get(k2);
4138
4139    Object r2b = sb.get(k2);
4140    int r1b = sb.size();
4141
4142    /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
          sb..size)" */
4143  }
4144
4145  }
4146
4147

```



```

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 =
4208         sb..size)" */
4209 }
4210 static void size_put_between_s_133(AssociationList sa, AssociationList sb, Object
4211     k2, Object v2)
4212 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
4213     sa..contents = sb..contents & sa..size = sb..size"
4214     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4215     ensures "True" */
4216 {
4217     int r1a = sa.size();
4218     /*: assume "EX v. (k2, v) : sa..contents" */
4219     Object r2a = sa.put(k2, v2);
4220
4221     Object r2b = sb.put(k2, v2);
4222     int r1b = sb.size();
4223
4224     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
4225         sb..size" */
4226 }
4227 static void size_put_between_c_133(AssociationList sa, AssociationList sb, Object
4228     k2, Object v2)
4229 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
4230     sa..contents = sb..contents & sa..size = sb..size"
4231     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4232     ensures "True" */
4233 {
4234     int r1a = sa.size();
4235     /*: assume "~(EX v. (k2, v) : sa..contents)" */
4236     Object r2a = sa.put(k2, v2);
4237
4238     Object r2b = sb.put(k2, v2);
4239     int r1b = sb.size();
4240
4241     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
4242         sb..size)" */
4243 }
4244 static void size_put_post_s_134(AssociationList sa, AssociationList sb, Object k2,
4245     Object v2)
4246 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
4247     sa..contents = sb..contents & sa..size = sb..size"
4248     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4249     ensures "True" */
4250 {
4251     int r1a = sa.size();
4252     Object r2a = sa.put(k2, v2);
4253     /*: assume "r2a ~= null" */
4254
4255     Object r2b = sb.put(k2, v2);
4256     int r1b = sb.size();
4257
4258     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
4259         sb..size" */
4260 }
4261 static void size_put_post_c_134(AssociationList sa, AssociationList sb, Object k2,
4262     Object v2)
4263 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
4264     sa..contents = sb..contents & sa..size = sb..size"
4265     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4266     ensures "True" */
4267 {

```

```

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(AssociationList sa, AssociationList sb, Object k2,
         Object v2)
4275 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
         sa..contents = sb..contents & sa..size = sb..size"
4276     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4277     ensures "True" */
4278 {
4279     /*: assume "EX v. (k2, v) : sa..contents" */
4280     int r1a = sa.size();
4281     sa.put(k2, v2);
4282
4283     sb.put(k2, v2);
4284     int r1b = sb.size();
4285
4286     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
4287 }
4288
4289 static void size_put_pre_c_135(AssociationList sa, AssociationList sb, Object k2,
         Object v2)
4290 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
         sa..contents = sb..contents & sa..size = sb..size"
4291     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4292     ensures "True" */
4293 {
4294     /*: assume "(EX v. (k2, v) : sa..contents)" */
4295     int r1a = sa.size();
4296     sa.put(k2, v2);
4297
4298     sb.put(k2, v2);
4299     int r1b = sb.size();
4300
4301     /*: assert "(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
4302 }
4303
4304 static void size_put_between_s_136(AssociationList sa, AssociationList sb, Object
         k2, Object v2)
4305 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
         sa..contents = sb..contents & sa..size = sb..size"
4306     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4307     ensures "True" */
4308 {
4309     int r1a = sa.size();
4310     /*: assume "EX v. (k2, v) : sa..contents" */
4311     sa.put(k2, v2);
4312
4313     sb.put(k2, v2);
4314     int r1b = sb.size();
4315
4316     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
4317 }
4318
4319 static void size_put_between_c_136(AssociationList sa, AssociationList sb, Object
         k2, Object v2)
4320 /*: 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,
4339     Object v2)
4340 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
4341     sa..contents = sb..contents & sa..size = sb..size"
4342     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4343     ensures "True" */
4344 {
4345     int r1a = sa.size();
4346     /*: ghost specvar sa__contents :: "(obj * obj) set" = "sa..contents" */
4347     sa.put(k2, v2);
4348     /*: assume "EX v. (k2, v) : sa__contents" */
4349
4350     sb.put(k2, v2);
4351     int r1b = sb.size();
4352
4353     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
4354 }
4355
4356 static void size_put_post_c_137(AssociationList sa, AssociationList sb, Object k2,
4357     Object v2)
4358 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null & v2 ~= null &
4359     sa..contents = sb..contents & sa..size = sb..size"
4360     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4361     ensures "True" */
4362 {
4363     int r1a = sa.size();
4364     /*: ghost specvar sa__contents :: "(obj * obj) set" = "sa..contents" */
4365     sa.put(k2, v2);
4366     /*: assume "~(EX v. (k2, v) : sa__contents)" */
4367
4368     sb.put(k2, v2);
4369     int r1b = sb.size();
4370
4371     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
4372 }
4373
4374 static void size_remove_pre_s_138(AssociationList sa, AssociationList sb, Object k2)
4375 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
4376     sa..contents = sb..contents & sa..size = sb..size"
4377     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4378     ensures "True" */
4379 {
4380     /*: assume "~(EX v. (k2, v) : sa..contents)" */
4381     int r1a = sa.size();
4382     Object r2a = sa.remove(k2);
4383
4384     Object r2b = sb.remove(k2);
4385     int r1b = sb.size();
4386
4387     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
4388         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 =
4402             sb..size)" */
4403 }
4404
4405 static void size_remove_between_s_139(AssociationList sa, AssociationList sb,
4406 Object k2)
4407 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
4408             sa..contents = sb..contents & sa..size = sb..size"
4409             modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4410             ensures "True" */
4411 {
4412     int r1a = sa.size();
4413     /*: assume "~(EX v. (k2, v) : sa..contents)" */
4414     Object r2a = sa.remove(k2);
4415
4416     Object r2b = sb.remove(k2);
4417     int r1b = sb.size();
4418
4419     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
4420             sb..size" */
4421 }
4422
4423 static void size_remove_between_c_139(AssociationList sa, AssociationList sb,
4424 Object k2)
4425 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
4426             sa..contents = sb..contents & sa..size = sb..size"
4427             modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4428             ensures "True" */
4429 {
4430     int r1a = sa.size();
4431     /*: assume "~(~(EX v. (k2, v) : sa..contents))" */
4432     Object r2a = sa.remove(k2);
4433
4434     Object r2b = sb.remove(k2);
4435     int r1b = sb.size();
4436
4437     /*: assert "(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
4438             sb..size)" */
4439 }
4440
4441 static void size_remove_post_s_140(AssociationList sa, AssociationList sb, Object
4442 k2)
4443 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
4444             sa..contents = sb..contents & sa..size = sb..size"
4445             modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4446             ensures "True" */
4447 {
4448     int r1a = sa.size();
4449     Object r2a = sa.remove(k2);
4450     /*: assume "r2a = null" */

```

```

4445     Object r2b = sb.remove(k2);
4446     int r1b = sb.size();
4447
4448     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
4449         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 =
4466         sb..size)" */
4467 }
4468
4469 static void size_remove_pre_s_141(AssociationList sa, AssociationList sb, Object k2)
4470 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
4471     sa..contents = sb..contents & sa..size = sb..size"
4472     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4473     ensures "True" */
4474 {
4475     /*: assume "~(EX v. (k2, v) : sa..contents)" */
4476     int r1a = sa.size();
4477     sa.remove(k2);
4478
4479     sb.remove(k2);
4480     int r1b = sb.size();
4481
4482     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
4483 }
4484
4485 static void size_remove_pre_c_141(AssociationList sa, AssociationList sb, Object k2)
4486 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
4487     sa..contents = sb..contents & sa..size = sb..size"
4488     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4489     ensures "True" */
4490 {
4491     /*: assume "~(~(EX v. (k2, v) : sa..contents))" */
4492     int r1a = sa.size();
4493     sa.remove(k2);
4494
4495     sb.remove(k2);
4496     int r1b = sb.size();
4497
4498     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
4499 }
4500
4501 static void size_remove_between_s_142(AssociationList sa, AssociationList sb,
    Object k2)
4502 /*: requires "sa ~= null & sb ~= null & sa ~= sb & k2 ~= null &
4503     sa..contents = sb..contents & sa..size = sb..size"
4504     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
4505     ensures "True" */
4506 {

```

```

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"

```



```

4629     ensures "True" */
4630 {
4631     int r1a = sa.size();
4632     int r2a = sa.size();
4633     /*: assume "True" */
4634
4635     int r2b = sb.size();
4636     int r1b = sb.size();
4637
4638     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
         sb..size" */
4639 }
4640
4641 static void size_size_post_c_146(AssociationList sa, AssociationList sb)
4642 /*: requires "sa ~= null & sb ~= null & sa ~= sb &
4643     sa..contents = sb..contents & sa..size = sb..size"
4644     ensures "True" */
4645 {
4646     int r1a = sa.size();
4647     int r2a = sa.size();
4648     /*: assume "~(True)" */
4649
4650     int r2b = sb.size();
4651     int r1b = sb.size();
4652
4653     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
         sb..size)" */
4654 }
4655 }
4656 }

```

### B.4.3 Inverse Testing Methods

Listing 12. AssociationListInv.java

```

1 class AssociationListInv {
2     static void put_0(AssociationList s, Object k, Object v)
3     /*: requires "s ~= null & k ~= null & v ~= null"
4         modifies "s..contents", "s..size"
5         ensures "True" */
6     {
7         Object r = s.put(k, v);
8         if (r != null) { s.put(k, r); } else { s.remove(k); }
9
10        /*: assert "s..contents = s..(old contents) & s..size = s..(old size)" */
11    }
12
13    static void remove_1(AssociationList s, Object k)
14    /*: requires "s ~= null & k ~= null"
15        modifies "s..contents", "s..size"
16        ensures "True" */
17    {
18        Object r = s.remove(k);
19        if (r != null) { s.put(k, r); }
20
21        /*: assert "s..contents = s..(old contents) & s..size = s..(old size)" */
22    }
23 }
24 }

```

## B.5 HashTable

### B.5.1 Data Structure

Listing 13. HashTable.java

```

1 public /*: claimedby HashTable */ class Node {
2     public Object key;
3     public Object value;
4     public Node next;
5
6     /*: public ghost specvar conts :: "(obj * obj) set" = "{}"
7     invariant ConAlloc: "ALL x y. (x,y) : conts --> x : alloc & y : alloc"
8     invariant ConNull: "null..conts = {}"
9     invariant ConDef: "this ~= null --> conts = {(key, value)} Un next..conts &
10    (ALL v. (key, v) ~: next..conts)"
11    invariant ConNonNull: "ALL x y. (x, y) : conts --> x ~= null & y ~= null" */
12 }
13 public class HashTable {
14     /*: public ghost specvar init :: "bool" = "False" */
15
16     private Node[] table = null;
17
18     /*: static specvar abs :: "(int => int)"
19     vardefs "abs == (%i1. if (i1 < 0) then (-i1) else i1)"
20
21     static specvar h :: "(obj => int => int)"
22     vardefs "h == (%o1. %i1. (abs (hashFunc o1)) mod i1)"
23     invariant HashInv: "init --> (ALL k. 0 <= (h k (table..length)) & (h k
24     (table..length)) < table..length)"
25
26     invariant TableNotNull: "init --> table ~= null"
27     invariant TableHidden: "init --> table : hidden"
28     invariant NodeHidden1: "init --> (ALL i. 0 <= i & i < table..length & table.[i]
29     ~= null --> table.[i] : hidden)"
30     invariant NodeHidden2: "ALL n. n : Node & n : alloc & n ~= null & n..next ~=
31     null --> n..next : hidden"
32     invariant FirstInjInv: "init --> (ALL i x y. y = x..next & y ~= null & 0 <= i &
33     i < table..length --> y ~= table.[i])"
34     invariant NextInjInv: "ALL x1 x2 y. y ~= null & y = x1..next & y = x2..next -->
35     x1 = x2"
36     invariant ElementInjInv: "init --> (ALL ht i j. ht : HashTable & ht : alloc &
37     ht..init & 0 <= i & i < ht..table..length & 0 <= j & j < table..length &
38     ht..table.[i] = table.[j] & table.[j] ~= null --> ht = this & i = j)"
39     invariant Coherence: "init --> (ALL i k v. 0 <= i & i < table..length --> (k,v)
40     : table.[i]..conts --> h k (table..length) = i)"
41     invariant TableInjInv: "ALL ht. ht..table = table & table ~= null --> ht = this"
42
43     public ghost specvar contents :: "(obj * obj) set" = "{}"
44     invariant ContentsDefInv: "init --> contents = {(k,v). (k,v) : table.[(h k
45     (table..length))].conts}"
46     public invariant MapInv: "ALL k v0 v1. (k,v0) : contents & (k,v1) : contents
47     --> v0 = v1" */
48
49     private int _size;
50
51     /*: public specvar size :: "int"
52     vardefs "size == _size"
53     invariant CardInv: "_size = card (contents)" */
54
55     public HashTable()
56     /*: modifies "contents", "size", "init"
57     ensures "init & contents = {} & size = 0" */
58     {
59         table = new /*: hidden */ Node[11];
60         /*: "contents" := "{}" */
61
62         _size = 0;
63
64     }
65 }

```

```

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
95     (table..length)) < table..length" from ArrayLength, HashFuncRel */
96     /*: note HashOther: "ALL ht. ht ~= this & ht : alloc & ht : HashTable &
97     ht..init --> (ALL k. 0 <= (h k (ht..table..length)) & (h k
98     (ht..table..length)) < ht..table..length)" from HashInv, NewNotHT,
99     TableInjInv */
100     /*: note ShowHashInv: "theinv HashInv" from HashThis, HashOther */
101   }
102   {
103     /*: pickAny x :: obj */
104     {
105       /*: assuming CardHyp: "x : alloc & x : HashTable" */
106       {
107         /*: assuming XIsThisHyp: "x = this" */
108         /*: note LengthZero: "x.._size = 0" */
109         /*: note ContentsEmpty: "x..contents = {}" */
110         /*: note XIsThisCard: "x.._size = card (x..contents)" from
111         XIsThisHyp, LengthZero, ContentsEmpty */
112       }
113       {
114         /*: assuming XNotThisHyp: "x ~= this" */
115         /*: note XInOldAlloc: "x : old alloc" */
116         /*: note OldCard: "x..(old HashTable._size) = card (x..(old
117         HashTable.contents))" from CardHyp, XNotThisHyp, XInOldAlloc,
118         CardInv */
119         /*: note XLengthEq: "x.._size = x..(old HashTable._size)" */
120         /*: note XContentsUnchanged: "x..contents = x..(old
121         HashTable.contents)" */
122         /*: note XNotThisCard: "x.._size = card (x..contents)" from
123         XLengthEq, OldCard, XContentsUnchanged */
124       }
125       /*: note CardConc: "x.._size = card (x..contents)" from XIsThisCard,
126       XNotThisCard */
127     }
128     /*: note CardPostCond: "x : alloc & x : HashTable --> x.._size = card
129     (x..contents)" forSuch x */
130   }
131 }
132
133 private int compute_hash(Object o1)
134 /*: requires "o1 ~= null & init & theinvs"
135 ensures "result = h o1 (table..length) & 0 <= result & result < table..length &
136 alloc = old alloc & theinvs" */
137 {
138   int hc = o1.hashCode();
139
140   if (hc < 0) { hc = -hc; }
141
142   /*: note LengthPos: "0 < table..length" */
143   /*: note ResLt: "(hc mod table..length) < table..length" from TrueBranch,
144   FalseBranch, LengthPos */
145
146   return (hc % table.length);
147 }
148
149 public boolean containsKey(Object k0)
150 /*: requires "init & k0 ~= null"
151 ensures "result = (EX v. ((k0, v) : contents))" */
152 {
153   return _containsKey(k0);
154 }
155
156 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
201    (table..length))].conts)}" from ContentsDefInv, ThisProps */
202  /*: note FNonNull: "f ~= null" */
203  /*: note FProps: "f : Node & f : alloc" from unalloc_lonely, array_pointsto,
    ThisProps */
204  /*: note VFound: "v0 = f..value" from InContents, OldContents, ConDef, KFound,
    FProps, FNonNull, HCProps */
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,
242            z) : x..(old HashTable.contents)" */
243         /*: note XContentsBack: "ALL y z. (y, z) : x..(old
244            HashTable.contents) --> (y, z) : x..contents" */
245         /*: note XContentsUnchanged: "x..contents = x..(old
246            HashTable.contents)" from XContentsForw, XContentsBack */
247     }
248     /*: note XNotThisCard: "x.._size = card (x..contents)" from
249        XLengthEq, OldCard, XContentsUnchanged */
250 }
251
252     /*: note CardConc: "x.._size = card (x..contents)" from XIsThisCard,
253        XNotThisCard */
254 }
255
256     /*: note CardPostCond: "x : alloc & x : HashTable --> x.._size = card
257        (x..contents)" forSuch x */
258 }
259
260     return f.value;
261 }
262
263 private Object removeFromBucket(Object k0, int hc)
264 /*: requires "comment ''Init'' init & k0 ~= null & (EX v.(k0, v) : contents) &
265    comment ''KNotFound'' (k0 ~= table.[hc]..key) & comment ''HCProps'' (0 <= hc &
266    hc < table..length & hc = h k0 (table..length)) & theinvs"
267    modifies "contents", "size", "conts", "next", "arrayState", "_size"
268    ensures "(k0, result) : old contents & contents = old contents - {(k0, result)}
269        & size = old size - 1 & (ALL a i. a ~= table --> a.[i] = old (a.[i])) &
270        theinvs" */
271 {
272     /*: ghost specvar v0 :: obj */
273     /*: havoc v0 suchThat InContents: "(k0, v0) : contents" */
274
275     Node f = table[hc];
276     Node prev = f;
277
278     /*: note InBucket: "(k0, v0) : prev..conts" from InContents, ContentsDefInv,
279        thisNotNull, thisType, Init, HCProps */
280     /*: note PrevNotNull: "prev ~= null" from InBucket, ConDef, ConNull */
281
282     /*: "prev..conts" := "prev..conts - {(k0, v0)}" */
283     /*: "contents" := "old contents - {(k0, v0)}" */
284
285     Node curr = prev.next;
286
287     /*: note PrevHidden: "prev : hidden" from NodeHidden1, thisNotNull, thisType,
288        PrevNotNull, Init, HCProps */
289     /*: note ConPreLoop: "ALL n. n : Node & n : alloc & n ~= null & n ~= prev -->
290        n..conts = {(n..key, n..value)} Un n..next..conts & (ALL v.(n..key, v) ~:
291        n..next..conts)" from ConDef, FirstInjInv, Init, HCProps, thisNotNull,
292        thisType, PrevNotNull */
293
294     /*: note ConUnchangedPreLoop: "ALL ht i. ht ~= this & ht : HashTable & ht :
295        alloc & ht..init & 0 <= i & i < ht..table..length --> ht..table.[i]..conts =
296        old (ht..table.[i]..conts)" from ElementInjInv, thisType, PrevNotNull, Init,
297        HCProps */
298
299     while /*: invariant "prev : Node & prev : alloc & prev ~= null & prev : hidden
300        & comment ''PrevCon'' (prev..conts = fieldRead (old conts) prev - {(k0,
301        v0)}) & comment ''PrevNot'' (ALL v.(prev..key, v) ~: prev..next..conts) &
302        comment ''CurrProps'' (curr : Node & curr : alloc) & comment ''CurrNotNull''
303        (curr ~= null) & comment ''PrevCurr'' (prev..next = curr & prev ~= curr) &
304        contents = old contents - {(k0, v0)} & (k0, v0) : curr..conts & comment
305        ''ConDefInv'' (ALL n. n : Node & n : alloc & n ~= null & n ~= prev -->
306        n..conts = {(n..key, n..value)} Un n..next..conts & (ALL v.(n..key, v) ~:
307        n..next..conts)) & comment ''ConLoop'' (ALL n. n..conts = old (n..conts) |

```

```

280 n..conts = old (n..conts) - {(k0, v0)} & (null..conts = {}) & comment
281 ''FConInv'' (f..conts = (fieldRead (old conts) f) - {(k0, v0)}) & comment
282 ''ConUnchanged'' (ALL ht i. ht ~= this & ht : HashTable & ht : alloc &
283 ht..init & 0 <= i & i < ht..table..length --> ht..table.[i]..conts = old
284 (ht..table.[i]..conts))" */ (curr.key != k0) {
285 /*: "curr..conts" := "curr..conts - {(k0, v0)}" */

286
287 /*: note CurrCon: "curr..conts = fieldRead (old conts) curr - {(k0, v0)}" */
288 /*: note PrevIsNot: "prev..key ~= k0" */
289 /*: note OldConDef: "fieldRead (old conts) prev = {(prev..key,
290 prev..value)} Un fieldRead (old conts) (prev..next)" */
291 /*: note PrevConDef: "prev..conts = {(prev..key, prev..value)} Un
292 prev..next..conts" from PrevCurr, PrevCon, CurrCon, OldConDef, PrevIsNot
293 */

294 prev = curr;
295 curr = curr.next;

296
297 /*: note FConLem: "f..conts = (fieldRead (old conts) f) - {(k0, v0)}" from
298 FConInv */

299
300 /*: note ConExceptPrev: "ALL n. n : Node & n : alloc & n ~= null & n ~=
301 prev --> n..conts = {(n..key, n..value)} Un n..next..conts & (ALL
302 v.(n..key, v) ~: n..next..conts)" from PrevConDef, PrevNot, ConDefInv,
303 PrevCurr, NextInjInv, CurrNotNull */
304 }

305 Node tmp = curr.next;
306 prev.next = tmp;
307 curr.next = null;

308
309 /*: "curr..conts" := "{(curr..key, curr..value)}" */

310 _size = _size - 1;

311 {
312 /*: pickAny x :: obj suchThat xHyp: "x : Node & x : alloc & x ~= null" */
313 {
314 /*: assuming xIsPrev: "x = prev" */
315 /*: note nextNotCurr: "fieldRead (old next) curr ~= curr" from
316 NextInjInv, CurrNotNull, PrevCurr, CurrProps */
317 /*: note prevNextCon: "prev..next..conts = fieldRead (old conts)
318 (prev..next)" */
319 /*: note prevOldCon: "fieldRead (old conts) prev = {(prev..key,
320 prev..value)} Un fieldRead (old conts) curr" */
321 /*: note currOldCon: "fieldRead (old conts) curr = {(curr..key,
322 curr..value)} Un fieldRead (old conts) (fieldRead (old next) curr)"
323 */
324 /*: note prevKeyNotK0: "prev..key ~= k0" */
325 {
326 /*: pickAny k :: obj, v :: obj suchThat ForwHyp: "(k, v) :
327 x..conts" */
328 /*: note kNotK0: "k ~= k0" */
329 /*: note currKeyIsK0: "curr..key = k0" */
330 /*: note ForwCase: "(k, v) : {(x..key, x..value)} Un
331 x..next..conts" from xHyp, xIsPrev, ForwHyp, PrevCurr,
332 nextNotCurr, PrevCon, prevNextCon, prevOldCon, currOldCon,
333 prevKeyNotK0, kNotK0, currKeyIsK0 forSuch k, v */
334 }
335 /*: note BackCase: "ALL k v. (k, v) : {(x..key, x..value)} Un
336 x..next..conts --> (k, v) : x..conts" */
337 /*: note xCon: "x..conts = {(x..key, x..value)} Un x..next..conts" from
338 ForwCase, BackCase */
339 }
340 }

```



```

321     /*: cases "x = curr", "x = prev", "x ~= curr & x ~= prev" for XCon:
322         "x..conts = {(x..key, x..value)} Un x..next..conts" */
323     /*: note ConPost: "x..conts = {(x..key, x..value)} Un x..next..conts & (ALL
324         v. (x..key, v) ~= x..next..conts)" forSuch x */
325 }
326 {
327     /*: pickAny ht :: obj suchThat CohHyp: "ht : alloc & ht : HashTable &
328         ht..init" */
329     {
330         /*: pickAny i :: int, k :: obj, v :: obj suchThat InnerHyp: "0 <= i & i
331             < ht..table..length & (k, v) : ht..table.[i]..conts" */
332         /*: note NotCurr: "ht..table.[i] ~= curr" */
333         /*: note InnerConc: "h k (ht..table..length) = i" from CohHyp,
334             InnerHyp, Coherence, NotCurr, ConLoop forSuch i, k, v */
335     }
336     /*: note CoherencePost: "(ALL i k v. 0 <= i & i < ht..table..length --> (k,
337         v) : ht..table.[i]..conts --> h k (ht..table..length) = i)" from
338         InnerConc forSuch ht */
339 }
340 {
341     /*: pickAny x :: obj suchThat ContentsDefHyp: "x : alloc & x : HashTable &
342         x..init" */
343     /*: note OldXContents: "fieldRead (old HashTable.contents) x = {(k, v). (k,
344         v) : fieldRead (old conts) (arrayRead (old arrayState) (x..table) (h k
345         (x..table..length)))}" from ContentsDefHyp, ContentsDefInv */
346     {
347         /*: assuming XNotThisHyp: "x ~= this" */
348         /*: note NotCurr: "ALL i. 0 <= i & i < x..table..length -->
349             x..table.[i] ~= curr" */
350         /*: note ConXUnchanged: "ALL i. 0 <= i & i < x..table..length -->
351             x..table.[i]..conts = fieldRead (old conts) (arrayRead (old
352             arrayState) (x..table) i)" from ContentsDefHyp, XNotThisHyp,
353             NotCurr, ConUnchanged */
354         /*: note LengthLemma: "ALL k. 0 <= (h k (x..table..length)) & (h k
355             (x..table..length)) < (x..table..length)" from ContentsDefHyp,
356             HashInv */
357         /*: note XNotThisCase: "x..contents = {(k, v). (k, v) : x..table.[(h k
358             (x..table..length))]..conts}" from XNotThisHyp, OldXContents,
359             LengthLemma, ConXUnchanged */
360     }
361     {
362         /*: assuming XIsThisHyp: "x = this" */
363         /*: note OldContents: "old contents = {(k,v). (k,v) : old (table.[(h k
364             (table..length))]..conts)}" */
365         {
366             /*: pickAny k :: obj, v :: obj suchThat ForwHyp: "(k, v) :
367                 contents" */
368             /*: note NotCurr: "table.[(h k (table..length))] ~= curr" from
369                 FirstInjInv, ContentsDefHyp, XIsThisHyp, PrevCurr, CurrNotNull,
370                 HashInv */
371             /*: note ForwCase: "(k, v) : table.[(h k (table..length))]..conts"
372                 from ForwHyp, OldContents, NotCurr, ConLoop forSuch k, v */
373         }
374         {
375             /*: pickAny k :: obj, v :: obj suchThat BackHyp: "(k, v) :
376                 table.[(h k (table..length))]..conts" */
377             /*: note NotCurr: "table.[(h k (table..length))] ~= curr" from
378                 FirstInjInv, ContentsDefHyp, XIsThisHyp, PrevCurr, CurrNotNull,
379                 HashInv */
380             /*: note BackCase: "(k, v) : contents" from BackHyp, OldContents,
381                 NotCurr, ConLoop, FConInv, HCProps forSuch k, v */
382         }
383     }
384 }

```

```

358     /*: note XIsThisCase: "contents = {(k, v). (k, v) : table.[(h k
359         (table..length))].conts}" from ForwCase, BackCase */
360     }
361     /*: note ContentsDefPost: "x..contents = {(k, v). (k, v) : x..table.[(h k
362         (x..table..length))].conts}" from XNotThisCase, XIsThisCase forSuch x */
363     }
364     /*: note ContentsPost: "contents = old contents - {(curr..key, curr..value)}" */
365     {
366     /*: pickAny x :: obj */
367     {
368     /*: assuming CardHyp: "x : alloc & x : HashTable" */
369     {
370     /*: note ThisProps: "this : old alloc & this : HashTable" */
371     /*: note OldCard: "old _size = card (old contents)" from ThisProps,
372         CardInv */
373     /*: note NewLength: "_size = old _size - 1" */
374     /*: note NewNotInOld: "(curr..key, curr..value) : old contents" */
375     /*: note XIsThisCard: "_size = card (contents)" from OldCard,
376         NewLength, NewNotInOld, ContentsPost */
377     }
378     {
379     /*: assuming XNotThisHyp: "x ~= this" */
380     /*: note XInOldAlloc: "x : old alloc" */
381     /*: note OldCard: "x..(old HashTable._size) = card (x..(old
382         HashTable.contents))" from CardHyp, XNotThisHyp, XInOldAlloc,
383         CardInv */
384     /*: note XLengthEq: "x.._size = x..(old HashTable._size)" */
385     {
386     /*: localize */
387     /*: note XContentsForw: "ALL y z. (y, z) : x..contents --> (y,
388         z) : x..(old HashTable.contents)" */
389     /*: note XContentsBack: "ALL y z. (y, z) : x..(old
390         HashTable.contents) --> (y, z) : x..contents" */
391     /*: note XContentsUnchanged: "x..contents = x..(old
392         HashTable.contents)" from XContentsForw, XContentsBack */
393     }
394     /*: note XNotThisCard: "x.._size = card (x..contents)" from
395         XLengthEq, OldCard, XContentsUnchanged */
396     }
397     /*: note CardConc: "x.._size = card (x..contents)" from XIsThisCard,
398         XNotThisCard */
399     }
400     /*: note CardPostCond: "x : alloc & x : HashTable --> x.._size = card
401         (x..contents)" forSuch x */
402     }
403     return curr.value;
404 }
405
406 private Object _remove(Object k0)
407 /*: requires "(comment ''Init'' init) & k0 ~= null & (EX v. (k0, v) : contents) &
408     theinvs"
409     modifies "contents", "size", "conts", "next", "arrayState", "_size"
410     ensures "(k0, result) : old contents & contents = old contents - {(k0, result)}
411         & size = old size - 1 & (ALL a i. a ~= table --> a.[i] = old (a.[i])) &
412         theinvs" */
413 {
414     /*: ghost specvar v0 :: obj */
415     /*: havoc v0 suchThat KeyInContents: "(k0, v0) : contents" */
416
417     int hc = compute_hash(k0);
418     Node f = table[hc];
419
420     /*: note ThisProps: "this : alloc & this : HashTable" */

```

```

408     /*: note HCDef: "hc = h k0 (table..length)" */
409     /*: note KeyInBucket: "(k0, v0) : table.[hc]..conts" from HCDef, KeyInContents,
        ContentsDef, Init, ThisProps */
410
411     if (f.key == k0) {
412         return removeFirst(k0, hc);
413     } else {
414         return removeFromBucket(k0, hc);
415     }
416 }
417
418 private void _add(Object k0, Object v0)
419 /*: requires "comment 'Init' init & k0 ~= null & v0 ~= null & ~(EX v. (k0, v) :
        contents) & theinvs"
420     modifies "contents", "size", "arrayState", "new..conts", "new..next",
        "new..value", "new..key", "_size"
421     ensures "contents = old contents Un {(k0, v0)} & size = old size + 1 & (ALL a
        i. a ~= table --> a.[i] = old (a.[i])) & theinvs" */
422 {
423     int hc = compute_hash(k0);
424     Node n = new /*: hidden */ Node();
425     n.key = k0;
426     n.value = v0;
427     Node first = table[hc];
428     n.next = first;
429     /*: "n..conts" := "{(k0, v0)} Un first..conts" */
430     table[hc] = n;
431     /*: "contents" := "(old contents) Un {(k0, v0)}" */
432
433     _size = _size + 1;
434
435     /*: note NewNotHT: "n ~: HashTable" */
436     /*: note ThisProps: "this : alloc & this : old alloc & this : HashTable" */
437     /*: note HCBounds: "0 <= hc & hc < table..length" */
438     /*: note NewOldNEq: "n ~= first" */
439
440     /*: note ThisTableNotNull: "table ~= null" */
441
442     /*: 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 */
443
444     /*: note HashPost: "theinv HashInv" from HashPost, HashInv, NewNotHT */
445
446     /*: note KeyAlloc: "k0 : alloc" */
447     /*: note ValAlloc: "v0 : alloc" */
448     /*: note AllocChanged: "alloc = old alloc Un {n}" */
449     /*: note FirstProps: "first : alloc & first : Node & first ~= n" from
        unalloc_lonely, AllocChanged, ThisProps, array_pointsto, NewNotHT */
450
451     /*: note ConAllocLemma: "theinv ConAlloc" from ConAllocLemma, ConAlloc,
        KeyAlloc, ValAlloc, FirstProps */
452
453     /*: note NewHidden2: "n..next ~= null --> n..next : hidden" from NewHidden2,
        NodeHidden1, ThisProps, Init, HCBounds */
454     /*: note OldHidden2: "ALL m. m ~= n & m : Node & m : alloc & m ~= null &
        m..next ~= null --> m..next : hidden" */
455     /*: note AllHidden2: "theinv NodeHidden2" from AllHidden2, NewHidden2,
        OldHidden2 */
456
457     /*: note NewHidden: "n : hidden" */
458     /*: 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
460  --> ht..table.[i] : hidden)" from OtherHidden1, NodeHidden1, NewNotHT */
      /*: note Hidden1All: "theinv NodeHidden1" from Hidden1All, ThisHidden1,
461  OtherHidden1 */
462
463  /*: note AllocChange: "Object.alloc = old Object.alloc Un {n}" */
464  /*: note HProp: "hc = h k0 (table..Array.length)" */
465
466  /*: note NewNotRefThisArr: "ALL i. 0 <= i & i < table..length --> (arrayRead
      (old arrayState) table i) ~= n" */
467  /*: note NewNotRefArray: "ALL a i. 0 <= i & i < a..length --> (arrayRead (old
468  arrayState) a i) ~= n" */
469
470  /*: note NewNotAlloc: "n ~: old alloc" */
471  /*: note NewNotRefByNext: "ALL x. x..next ~= n" from NewOldNEq, NewNotAlloc,
472  unalloc_lonely, NewNotRefByNext */
473
474  /*: note NotInOldContents: "ALL v. (k0, v) ~: old contents" */
475  /*: note NotInOldConFirst: "ALL v. (k0, v) ~: (fieldRead (old conts) first)"
476  from NotInOldConFirst, NotInOldContents, ContentsDef, ThisProps, Init, HProp
477  */
478  /*: note FirstNotN: "first ~= n" */
479  /*: note NewConDef: "theinv ConDef" from NewConDef, ConDef, NewNotRefByNext,
480  FirstNotN, NotInOldConFirst */
481
482  /*: note ElemInj: "theinv ElementInjInv" from ElementInjInv, ThisProps,
483  NewNotHT, NewNotRefArray, TableInjInv, ThisTableNotNull */
484
485  {
486    /*: pickAny ht :: obj */
487    {
488      /*: assuming h1: "ht : alloc & ht : HashTable & ht..init" */
489      {
490        /*: pickAny i :: int, k :: obj, v :: obj */
491        /*: note g1: "arrayRead (old arrayState) (ht..table) i ~= n" */
492        /*: note g2: "ht : old alloc" */
493        {
494          /*: assuming h2: "0 <= i & i < ht..table..length & (k,v) :
495          ht..table.[i]..conts" */
496          {
497            /*: assuming h3: "ht = this" */
498            /*: note c5: "h k (ht..table..length) = i" from c3, h1, h2,
499            h3, Coherence, g1, g2, HProp */
500          }
501          {
502            /*: assuming h4: "ht ~= this" */
503            /*: note g3: "ht..table ~= table" */
504            /*: note c4: "h k (ht..table..length) = i" from c4, h1, h2,
505            h4, Coherence, g1, g2, g3 */
506          }
507          /*: note c3: "h k (ht..table..length) = i" from c4, c5 */
508        }
509        /*: note c2: "0 <= i & i < ht..table..length --> (k,v) :
510        ht..table.[i]..conts --> h k (ht..table..length) = i" forSuch i,
511        k, v */
512      }
513      /*: note c1: "ALL i k v. 0 <= i & i < ht..table..length --> (k,v) :
514      ht..table.[i]..conts --> h k (ht..table..length) = i" */
515    }
516    /*: note CohPost: "ht : alloc & ht : HashTable & ht..init --> (ALL i k v. 0
517    <= i & i < ht..table..length --> (k,v) : ht..table.[i]..conts --> h k
518    (ht..table..length) = i)" forSuch ht */
519  }
520
521 }
522
523 }
524
525 }

```

```

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
511       (table..length))].conts}" from ThisContents, HProp,
512       NewNotRefThisArr, HashInv, ContentsDefInv, ThisProps, Init */
513     {
514       /*: assuming OtherHyp: "ht ~= this" */
515       /*: note OtherContents: "ht..contents = {(k,v). (k,v) :
516         ht..table.[(h k (ht..table..length))].conts}" from OtherHyp,
517         NewContentsHyp, ContentsDefInvHash, NewNotHT, TableInjInvHash,
518         NewNotRefArray, TableNotNullHash, HashInv */
519     }
520     /*: cases "ht = this", "ht ~= this" for NewContentsCases: "ht..contents
521       = {(k,v). (k,v) : ht..table.[(h k (ht..table..length))].conts}"
522       from ThisContents, OtherContents */
523     /*: note NewContentsConc: "ht..contents = {(k,v). (k,v) : ht..table.[(h
524       k (ht..table..length))].conts}" */
525   }
526   /*: note NewContentsDef: "ht : alloc & ht : HashTable & ht..init -->
527     ht..contents = {(k,v). (k,v) : ht..table.[(h k
528       (ht..table..length))].conts}" forSuch ht */
529 }
530
531 /*: note OldNotRefByNext: "ALL x. first ~= null --> old (x..next) ~= first"
532 from FirstInjInv, Init, HCBounds, OldNotRefByNext, ThisProps */
533
534 /*: note NewFirstInj: "theinv FirstInjInv" from FirstInjInv, NewNotRefByNext,
535 OldNotRefByNext, TableInjInv, OldNotRefInTable, HCBounds, NewFirstInj,
536 ThisProps, ElementInjInv, NewNotHT */
537
538 /*: note NewNextInj: "theinv NextInjInv" from NextInjInv, OldNotRefByNext,
539 NewNextInj */
540
541 /*: note ContentsPost: "contents = old contents Un {(k0, v0)}" */
542 {
543   /*: pickAny x :: obj */
544   {
545     /*: assuming CardHyp: "x : alloc & x : HashTable" */
546     {
547       /*: note ThisProps: "this : old alloc & this : HashTable" */
548       /*: note OldCard: "old _size = card (old contents)" from ThisProps,
549         CardInv */
550       /*: note NewLength: "_size = old _size + 1" */
551       /*: note NewNotInOld: "(k0, v0) ~: old contents" */
552       /*: note XIsThisCard: "_size = card (contents)" from OldCard,
553         NewLength, NewNotInOld, ContentsPost */
554     }
555     {
556       /*: assuming XNotThisHyp: "x ~= this" */
557       /*: note XInOldAlloc: "x : old alloc" */
558       /*: note OldCard: "x..(old HashTable._size) = card (x..(old
559         HashTable.contents))" from CardHyp, XNotThisHyp, XInOldAlloc,
560         CardInv */
561       /*: note XLengthEq: "x.._size = x..(old HashTable._size)" */
562       {
563         /*: localize */
564         /*: note XContentsForw: "ALL y z. (y, z) : x..contents --> (y,
565           z) : x..(old HashTable.contents)" */
566         /*: note XContentsBack: "ALL y z. (y, z) : x..(old
567           HashTable.contents) --> (y, z) : x..contents" */
568         /*: note XContentsUnchanged: "x..contents = x..(old
569           HashTable.contents)" from XContentsForw, XContentsBack */
570       }
571     }
572   }
573 }

```

```

550         }
551         /*: note XNotThisCard: "x.._size = card (x..contents)" from
           XLengthEq, OldCard, XContentsUnchanged */
552     }
553     /*: note CardConc: "x.._size = card (x..contents)" from XIsThisCard,
           XNotThisCard */
554     }
555     /*: note CardPostCond: "x : alloc & x : HashTable --> x.._size = card
           (x..contents)" forSuch x */
556     }
557 }
558
559 public Object put(Object k0, Object v0)
560 /*: requires "init & k0 ~= null & v0 ~= null"
561    modifies "contents", "size"
562    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)"
           */
563 {
564     if (_containsKey(k0)) {
565         Object v1 = _remove(k0);
566         _add(k0, v0);
567         return v1;
568     } else {
569         _add(k0, v0);
570         return null;
571     }
572 }
573
574 public Object get(Object k0)
575 /*: requires "init & k0 ~= null"
576    ensures "(result ~= null --> (k0, result) : contents) & (result = null --> ~(EX
           v. (k0, v) : contents))" */
577 {
578     /*: instantiate "theinv ContentsDefInv" with "this" */
579     /*: mp ThisContentsDef: "this : alloc & this : HashTable & init --> contents =
           {(k, v). (k, v) : table.[(h k (table..length))].conts}" */
580     int hc = compute_hash(k0);
581     Node curr = table[hc];
582
583     /*: note HCDef: "hc = h k0 (table..length)" */
584     /*: note InCurr: "ALL v. ((k0, v) : contents) = ((k0, v) : curr..conts)" from
           ThisContentsDef, HCDef */
585
586     while /*: invariant "ALL v. ((k0, v) : contents) = ((k0, v) : curr..conts)" */
           (curr != null) {
587         if (curr.key == k0) {
588             return curr.value;
589         }
590
591         curr = curr.next;
592     }
593     return null;
594 }
595
596 public int size()
597 /*: ensures "result = size" */
598 {
599     return _size;
600 }

```

## B.5.2 Commutativity Testing Methods

Listing 14. HashTableComm.java

```

1 class HashTableComm {
2     static void containsKey_containsKey_pre_s_0(HashTable sa, HashTable sb, Object k1,
3         Object k2)
4     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
5         & k2 ~= null &
6             sa..contents = sb..contents & sa..size = sb..size"
7         ensures "True" */
8     {
9         /*: assume "True" */
10        boolean r1a = sa.containsKey(k1);
11        boolean r2a = sa.containsKey(k2);
12
13        boolean r2b = sb.containsKey(k2);
14        boolean r1b = sb.containsKey(k1);
15
16        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
17            sb..size" */
18    }
19
20    static void containsKey_containsKey_pre_c_0(HashTable sa, HashTable sb, Object k1,
21        Object k2)
22    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
23        & k2 ~= null &
24            sa..contents = sb..contents & sa..size = sb..size"
25        ensures "True" */
26    {
27        /*: assume "~(True)" */
28        boolean r1a = sa.containsKey(k1);
29        boolean r2a = sa.containsKey(k2);
30
31        boolean r2b = sb.containsKey(k2);
32        boolean r1b = sb.containsKey(k1);
33
34        /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
35            sb..size)" */
36    }
37
38    static void containsKey_containsKey_between_s_1(HashTable sa, HashTable sb, Object
39        k1, Object k2)
40    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
41        & k2 ~= null &
42            sa..contents = sb..contents & sa..size = sb..size"
43        ensures "True" */
44    {
45        boolean r1a = sa.containsKey(k1);
46        /*: assume "True" */
47        boolean r2a = sa.containsKey(k2);
48
49        boolean r2b = sb.containsKey(k2);
50        boolean r1b = sb.containsKey(k1);
51
52        /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
53            sb..size" */
54    }
55
56    static void containsKey_containsKey_between_c_1(HashTable sa, HashTable sb, Object
57        k1, Object k2)
58    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
59        & 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 =
60         sb..size)" */
61 }
62
63 static void containsKey_containsKey_post_s_2(HashTable sa, HashTable sb, Object k1,
64     Object k2)
65 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
66     & k2 ~= null &
67     sa..contents = sb..contents & sa..size = sb..size"
68     ensures "True" */
69 {
70     boolean r1a = sa.containsKey(k1);
71     boolean r2a = sa.containsKey(k2);
72     /*: assume "True" */
73
74     boolean r2b = sb.containsKey(k2);
75     boolean r1b = sb.containsKey(k1);
76
77     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
78         sb..size" */
79 }
80
81 static void containsKey_containsKey_post_c_2(HashTable sa, HashTable sb, Object k1,
82     Object k2)
83 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
84     & k2 ~= null &
85     sa..contents = sb..contents & sa..size = sb..size"
86     ensures "True" */
87 {
88     boolean r1a = sa.containsKey(k1);
89     boolean r2a = sa.containsKey(k2);
90     /*: assume "~(True)" */
91
92     boolean r2b = sb.containsKey(k2);
93     boolean r1b = sb.containsKey(k1);
94
95     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
96         sb..size)" */
97 }
98
99 static void containsKey_get_pre_s_3(HashTable sa, HashTable sb, Object k1, Object
100     k2)
101 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
102     & k2 ~= null &
103     sa..contents = sb..contents & sa..size = sb..size"
104     ensures "True" */
105 {
106     /*: assume "True" */
107     boolean r1a = sa.containsKey(k1);
108     Object r2a = sa.get(k2);
109
110     Object r2b = sb.get(k2);
111     boolean r1b = sb.containsKey(k1);

```



```

104     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
105         sb..size" */
106 }
107 static void containsKey_get_pre_c_3(HashTable sa, HashTable sb, Object k1, Object
108     k2)
109 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
110     & k2 ~= null &
111         sa..contents = sb..contents & sa..size = sb..size"
112     ensures "True" */
113 {
114     /*: assume "~(True)" */
115     boolean r1a = sa.containsKey(k1);
116     Object r2a = sa.get(k2);
117
118     Object r2b = sb.get(k2);
119     boolean r1b = sb.containsKey(k1);
120
121     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
122         sb..size)" */
123 }
124 static void containsKey_get_between_s_4(HashTable sa, HashTable sb, Object k1,
125     Object k2)
126 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
127     & k2 ~= null &
128         sa..contents = sb..contents & sa..size = sb..size"
129     ensures "True" */
130 {
131     boolean r1a = sa.containsKey(k1);
132     /*: assume "True" */
133     Object r2a = sa.get(k2);
134
135     Object r2b = sb.get(k2);
136     boolean r1b = sb.containsKey(k1);
137
138     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
139         sb..size" */
140 }
141 static void containsKey_get_between_c_4(HashTable sa, HashTable sb, Object k1,
142     Object k2)
143 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
144     & k2 ~= null &
145         sa..contents = sb..contents & sa..size = sb..size"
146     ensures "True" */
147 {
148     boolean r1a = sa.containsKey(k1);
149     /*: assume "~(True)" */
150     Object r2a = sa.get(k2);
151
152     Object r2b = sb.get(k2);
153     boolean r1b = sb.containsKey(k1);
154
155     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
156         sb..size)" */
157 }
158 static void containsKey_get_post_s_5(HashTable sa, HashTable sb, Object k1, Object
159     k2)
160 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
161     & k2 ~= null &
162         sa..contents = sb..contents & sa..size = sb..size"
163     ensures "True" */
164 {

```

```

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 =
165         sb..size" */
166 }
167
168 static void containsKey_get_post_c_5(HashTable sa, HashTable sb, Object k1, Object
169     k2)
170 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
171     & k2 ~= null &
172         sa..contents = sb..contents & sa..size = sb..size"
173     ensures "True" */
174 {
175     boolean r1a = sa.containsKey(k1);
176     Object r2a = sa.get(k2);
177     /*: assume "~(True)" */
178
179     Object r2b = sb.get(k2);
180     boolean r1b = sb.containsKey(k1);
181
182     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
183         sb..size)" */
184 }
185
186 static void containsKey_put_pre_s_6(HashTable sa, HashTable sb, Object k1, Object
187     k2, Object v2)
188 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
189     & k2 ~= null & v2 ~= null &
190         sa..contents = sb..contents & sa..size = sb..size"
191     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
192     ensures "True" */
193 {
194     /*: assume "k1 ~= k2 | (EX v. (k1, v) : sa..contents)" */
195     boolean r1a = sa.containsKey(k1);
196     Object r2a = sa.put(k2, v2);
197
198     Object r2b = sb.put(k2, v2);
199     boolean r1b = sb.containsKey(k1);
200
201     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
202         sb..size" */
203 }
204
205 static void containsKey_put_pre_c_6(HashTable sa, HashTable sb, Object k1, Object
206     k2, Object v2)
207 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
208     & k2 ~= null & v2 ~= null &
209         sa..contents = sb..contents & sa..size = sb..size"
210     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
211     ensures "True" */
212 {
213     /*: assume "~(k1 ~= k2 | (EX v. (k1, v) : sa..contents))" */
214     boolean r1a = sa.containsKey(k1);
215     Object r2a = sa.put(k2, v2);
216
217     Object r2b = sb.put(k2, v2);
218     boolean r1b = sb.containsKey(k1);
219
220     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
221         sb..size)" */

```

```

212 }
213
214 static void containsKey_put_between_s_7(HashTable sa, HashTable sb, Object k1,
215     Object k2, Object v2)
216 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
217 & k2 ~= null & v2 ~= null &
218     sa..contents = sb..contents & sa..size = sb..size"
219 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
220 ensures "True" */
221 {
222     boolean r1a = sa.containsKey(k1);
223     /*: assume "k1 ~= k2 | r1a" */
224     Object r2a = sa.put(k2, v2);
225
226     Object r2b = sb.put(k2, v2);
227     boolean r1b = sb.containsKey(k1);
228
229     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
230     sb..size" */
231 }
232
233 static void containsKey_put_between_c_7(HashTable sa, HashTable sb, Object k1,
234     Object k2, Object v2)
235 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
236 & k2 ~= null & v2 ~= null &
237     sa..contents = sb..contents & sa..size = sb..size"
238 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
239 ensures "True" */
240 {
241     boolean r1a = sa.containsKey(k1);
242     /*: assume "~(k1 ~= k2 | r1a)" */
243     Object r2a = sa.put(k2, v2);
244
245     Object r2b = sb.put(k2, v2);
246     boolean r1b = sb.containsKey(k1);
247
248     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
249     sb..size)" */
250 }
251
252 static void containsKey_put_post_s_8(HashTable sa, HashTable sb, Object k1, Object
253     k2, Object v2)
254 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
255 & k2 ~= null & v2 ~= null &
256     sa..contents = sb..contents & sa..size = sb..size"
257 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
258 ensures "True" */
259 {
260     boolean r1a = sa.containsKey(k1);
261     Object r2a = sa.put(k2, v2);
262     /*: assume "k1 ~= k2 | r1a" */
263
264     Object r2b = sb.put(k2, v2);
265     boolean r1b = sb.containsKey(k1);
266
267     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
268     sb..size" */
269 }
270
271 static void containsKey_put_post_c_8(HashTable sa, HashTable sb, Object k1, Object
272     k2, Object v2)
273 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
274 & k2 ~= null & v2 ~= null &
275     sa..contents = sb..contents & sa..size = sb..size"
276 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,
327     Object k2, Object v2)
328 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
329     & k2 ~= null & v2 ~= null &
330     sa..contents = sb..contents & sa..size = sb..size"
331     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
332     ensures "True" */
333 {
334     boolean r1a = sa.containsKey(k1);
335     /*: assume "~(k1 ~= k2 | r1a)" */
336     sa.put(k2, v2);
337
338     sb.put(k2, v2);
339     boolean r1b = sb.containsKey(k1);
340
341     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
342 }
343
344 static void containsKey_put_post_s_11(HashTable sa, HashTable sb, Object k1, Object
345     k2, Object v2)
346 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
347     & k2 ~= null & v2 ~= null &
348     sa..contents = sb..contents & sa..size = sb..size"
349     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
350     ensures "True" */
351 {
352     boolean r1a = sa.containsKey(k1);
353     sa.put(k2, v2);
354     /*: assume "k1 ~= k2 | r1a" */
355
356     sb.put(k2, v2);
357     boolean r1b = sb.containsKey(k1);
358
359     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
360 }
361
362 static void containsKey_put_post_c_11(HashTable sa, HashTable sb, Object k1, Object
363     k2, Object v2)
364 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
365     & k2 ~= null & v2 ~= null &
366     sa..contents = sb..contents & sa..size = sb..size"
367     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
368     ensures "True" */
369 {
370     boolean r1a = sa.containsKey(k1);
371     sa.put(k2, v2);
372     /*: assume "~(k1 ~= k2 | r1a)" */
373
374     sb.put(k2, v2);
375     boolean r1b = sb.containsKey(k1);
376
377     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
378 }
379
380 static void containsKey_remove_pre_s_12(HashTable sa, HashTable sb, Object k1,
381     Object k2)
382 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
383     & k2 ~= null &
384     sa..contents = sb..contents & sa..size = sb..size"
385     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
386     ensures "True" */
387 {
388     /*: 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,
439 Object k2)
440 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
441 & k2 ~= null &
442 sa..contents = sb..contents & sa..size = sb..size"
443 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
444 ensures "True" */
445 {
446 boolean r1a = sa.containsKey(k1);
447 Object r2a = sa.remove(k2);
448 /*: assume "k1 ~= k2 | ~r1a" */
449
450 Object r2b = sb.remove(k2);
451 boolean r1b = sb.containsKey(k1);
452
453 /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
454 sb..size" */
455 }
456
457 static void containsKey_remove_post_c_14(HashTable sa, HashTable sb, Object k1,
458 Object k2)
459 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
460 & k2 ~= null &
461 sa..contents = sb..contents & sa..size = sb..size"
462 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
463 ensures "True" */
464 {
465 boolean r1a = sa.containsKey(k1);
466 Object r2a = sa.remove(k2);
467 /*: assume "~(k1 ~= k2 | ~r1a)" */
468
469 Object r2b = sb.remove(k2);
470 boolean r1b = sb.containsKey(k1);
471
472 /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
473 sb..size)" */
474 }
475
476 static void containsKey_remove_pre_s_15(HashTable sa, HashTable sb, Object k1,
477 Object k2)
478 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
479 & k2 ~= null &
480 sa..contents = sb..contents & sa..size = sb..size"
481 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
482 ensures "True" */
483 {
484 /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
485 boolean r1a = sa.containsKey(k1);
486 sa.remove(k2);
487
488 sb.remove(k2);
489 boolean r1b = sb.containsKey(k1);
490
491 /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
492 }
493
494 static void containsKey_remove_pre_c_15(HashTable sa, HashTable sb, Object k1,
495 Object k2)
496 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
497 & k2 ~= null &
498 sa..contents = sb..contents & sa..size = sb..size"
499 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
500 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,
503     Object k2)
504 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
505     & k2 ~= null &
506     sa..contents = sb..contents & sa..size = sb..size"
507     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
508     ensures "True" */
509 {
510     boolean r1a = sa.containsKey(k1);
511     /*: assume "k1 ~= k2 | ~r1a" */
512     sa.remove(k2);
513
514     sb.remove(k2);
515     boolean r1b = sb.containsKey(k1);
516
517     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
518 }
519
520 static void containsKey_remove_between_c_16(HashTable sa, HashTable sb, Object k1,
521     Object k2)
522 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
523     & k2 ~= null &
524     sa..contents = sb..contents & sa..size = sb..size"
525     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
526     ensures "True" */
527 {
528     boolean r1a = sa.containsKey(k1);
529     /*: assume "~(k1 ~= k2 | ~r1a)" */
530     sa.remove(k2);
531
532     sb.remove(k2);
533     boolean r1b = sb.containsKey(k1);
534
535     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
536 }
537
538 static void containsKey_remove_post_s_17(HashTable sa, HashTable sb, Object k1,
539     Object k2)
540 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
541     & k2 ~= null &
542     sa..contents = sb..contents & sa..size = sb..size"
543     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
544     ensures "True" */
545 {
546     boolean r1a = sa.containsKey(k1);
547     sa.remove(k2);
548     /*: assume "k1 ~= k2 | ~r1a" */
549
550     sb.remove(k2);
551     boolean r1b = sb.containsKey(k1);
552
553     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
554 }

```



```

550 static void containsKey_remove_post_c_17(HashTable sa, HashTable sb, Object k1,
551 Object k2)
552 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
553 & k2 ~= null &
554 sa..contents = sb..contents & sa..size = sb..size"
555 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
556 ensures "True" */
557 {
558 boolean r1a = sa.containsKey(k1);
559 sa.remove(k2);
560 /*: assume "~(k1 ~= k2 | ~r1a)" */
561 sb.remove(k2);
562 boolean r1b = sb.containsKey(k1);
563 /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
564 }
565
566 static void containsKey_size_pre_s_18(HashTable sa, HashTable sb, Object k1)
567 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
568 &
569 sa..contents = sb..contents & sa..size = sb..size"
570 ensures "True" */
571 {
572 /*: assume "True" */
573 boolean r1a = sa.containsKey(k1);
574 int r2a = sa.size();
575
576 int r2b = sb.size();
577 boolean r1b = sb.containsKey(k1);
578
579 /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
580 sb..size" */
581 }
582
583 static void containsKey_size_pre_c_18(HashTable sa, HashTable sb, Object k1)
584 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
585 &
586 sa..contents = sb..contents & sa..size = sb..size"
587 ensures "True" */
588 {
589 /*: assume "~(True)" */
590 boolean r1a = sa.containsKey(k1);
591 int r2a = sa.size();
592
593 int r2b = sb.size();
594 boolean r1b = sb.containsKey(k1);
595
596 /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
597 sb..size)" */
598 }
599
600 static void containsKey_size_between_s_19(HashTable sa, HashTable sb, Object k1)
601 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
602 &
603 sa..contents = sb..contents & sa..size = sb..size"
604 ensures "True" */
605 {
606 boolean r1a = sa.containsKey(k1);
607 /*: assume "True" */
608 int r2a = sa.size();
609
610 int r2b = sb.size();
611 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         boolean r2b = sb.containsKey(k2);
665         Object r1b = sb.get(k1);
666
667         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
668             sb..size" */
669     }
670
671     static void get_containsKey_pre_c_21(HashTable sa, HashTable sb, Object k1, Object
672         k2)
673     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
674         & k2 ~= null &
675             sa..contents = sb..contents & sa..size = sb..size"
676         ensures "True" */
677     {
678         /*: assume "~(True)" */
679         Object r1a = sa.get(k1);
680         boolean r2a = sa.containsKey(k2);
681
682         boolean r2b = sb.containsKey(k2);
683         Object r1b = sb.get(k1);
684
685         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
686             sb..size)" */
687     }
688
689     static void get_containsKey_between_s_22(HashTable sa, HashTable sb, Object k1,
690         Object k2)
691     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
692         & k2 ~= null &
693             sa..contents = sb..contents & sa..size = sb..size"
694         ensures "True" */
695     {
696         Object r1a = sa.get(k1);
697         /*: assume "True" */
698         boolean r2a = sa.containsKey(k2);
699
700         boolean r2b = sb.containsKey(k2);
701         Object r1b = sb.get(k1);
702
703         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
704             sb..size" */
705     }
706
707     static void get_containsKey_between_c_22(HashTable sa, HashTable sb, Object k1,
708         Object k2)
709     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
710         & k2 ~= null &
711             sa..contents = sb..contents & sa..size = sb..size"
712         ensures "True" */
713     {
714         Object r1a = sa.get(k1);
715         /*: assume "~(True)" */
716         boolean r2a = sa.containsKey(k2);
717
718         boolean r2b = sb.containsKey(k2);
719         Object r1b = sb.get(k1);
720
721         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
722             sb..size)" */
723     }
724
725     static void get_containsKey_post_s_23(HashTable sa, HashTable sb, Object k1, Object
726         k2)

```

```

717  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
718      & k2 ~= null &
719      sa..contents = sb..contents & sa..size = sb..size"
720      ensures "True" */
721  {
722      Object r1a = sa.get(k1);
723      boolean r2a = sa.containsKey(k2);
724      /*: assume "True" */
725
726      boolean r2b = sb.containsKey(k2);
727      Object r1b = sb.get(k1);
728
729      /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
730      sb..size" */
731  }
732
733  static void get_containsKey_post_c_23(HashTable sa, HashTable sb, Object k1, Object
734      k2)
735  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
736      & k2 ~= null &
737      sa..contents = sb..contents & sa..size = sb..size"
738      ensures "True" */
739  {
740      Object r1a = sa.get(k1);
741      boolean r2a = sa.containsKey(k2);
742      /*: assume "~(True)" */
743
744      boolean r2b = sb.containsKey(k2);
745      Object r1b = sb.get(k1);
746
747      /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
748      sb..size)" */
749  }
750
751  static void get_get_pre_s_24(HashTable sa, HashTable sb, Object k1, Object k2)
752  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
753      & k2 ~= null &
754      sa..contents = sb..contents & sa..size = sb..size"
755      ensures "True" */
756  {
757      /*: assume "True" */
758      Object r1a = sa.get(k1);
759      Object r2a = sa.get(k2);
760
761      Object r2b = sb.get(k2);
762      Object r1b = sb.get(k1);
763
764      /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
765      sb..size" */
766  }
767
768  static void get_get_pre_c_24(HashTable sa, HashTable sb, Object k1, Object k2)
769  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
770      & k2 ~= null &
771      sa..contents = sb..contents & sa..size = sb..size"
772      ensures "True" */
773  {
774      /*: assume "~(True)" */
775      Object r1a = sa.get(k1);
776      Object r2a = sa.get(k2);
777
778      Object r2b = sb.get(k2);
779      Object r1b = sb.get(k1);

```

```

773     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
774         sb..size)" */
775 }
776 static void get_get_between_s_25(HashTable sa, HashTable sb, Object k1, Object k2)
777 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
778     & k2 ~= null &
779         sa..contents = sb..contents & sa..size = sb..size"
780     ensures "True" */
781 {
782     Object r1a = sa.get(k1);
783     /*: assume "True" */
784     Object r2a = sa.get(k2);
785
786     Object r2b = sb.get(k2);
787     Object r1b = sb.get(k1);
788
789     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
790         sb..size" */
791 }
792 static void get_get_between_c_25(HashTable sa, HashTable sb, Object k1, Object k2)
793 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
794     & k2 ~= null &
795         sa..contents = sb..contents & sa..size = sb..size"
796     ensures "True" */
797 {
798     Object r1a = sa.get(k1);
799     /*: assume "~(True)" */
800     Object r2a = sa.get(k2);
801
802     Object r2b = sb.get(k2);
803     Object r1b = sb.get(k1);
804
805     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
806         sb..size)" */
807 }
808 static void get_get_post_s_26(HashTable sa, HashTable sb, Object k1, Object k2)
809 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
810     & k2 ~= null &
811         sa..contents = sb..contents & sa..size = sb..size"
812     ensures "True" */
813 {
814     Object r1a = sa.get(k1);
815     Object r2a = sa.get(k2);
816     /*: assume "True" */
817
818     Object r2b = sb.get(k2);
819     Object r1b = sb.get(k1);
820
821     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
822         sb..size" */
823 }
824 static void get_get_post_c_26(HashTable sa, HashTable sb, Object k1, Object k2)
825 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
826     & k2 ~= null &
827         sa..contents = sb..contents & sa..size = sb..size"
828     ensures "True" */
829 {
830     Object r1a = sa.get(k1);
831     Object r2a = sa.get(k2);
832     /*: assume "~(True)" */

```

```

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(HashTable sa, HashTable sb, Object k1, Object k2,
      Object v2)
837 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashTable sa, HashTable sb, Object k1, Object k2,
      Object v2)
853 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashTable sa, HashTable sb, Object k1, Object k2,
      Object v2)
869 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(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,
949     Object v2)
950 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
951 & k2 ~= null & v2 ~= null &
952     sa..contents = sb..contents & sa..size = sb..size"
953 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
954 ensures "True" */
955 {
956     /*: assume "~(k1 ~= k2 | (k1, v2) : sa..contents)" */
957     Object r1a = sa.get(k1);
958     sa.put(k2, v2);
959
960     sb.put(k2, v2);
961     Object r1b = sb.get(k1);
962
963     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
964 }
965
966 static void get_put_between_s_31(HashTable sa, HashTable sb, Object k1, Object k2,
967     Object v2)
968 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
969 & k2 ~= null & v2 ~= null &
970     sa..contents = sb..contents & sa..size = sb..size"
971 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
972 ensures "True" */
973 {
974     Object r1a = sa.get(k1);
975     /*: assume "k1 ~= k2 | r1a = v2" */
976     sa.put(k2, v2);
977
978     sb.put(k2, v2);
979     Object r1b = sb.get(k1);
980
981     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
982 }
983
984 static void get_put_between_c_31(HashTable sa, HashTable sb, Object k1, Object k2,
985     Object v2)
986 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
987 & k2 ~= null & v2 ~= null &
988     sa..contents = sb..contents & sa..size = sb..size"
989 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
990 ensures "True" */
991 {
992     Object r1a = sa.get(k1);
993     /*: assume "~(k1 ~= k2 | r1a = v2)" */
994     sa.put(k2, v2);
995
996     sb.put(k2, v2);
997     Object r1b = sb.get(k1);
998
999     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1000 }

```



```

996 static void get_put_post_s_32(HashTable sa, HashTable sb, Object k1, Object k2,
997     Object v2)
998 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
999     & k2 ~= null & v2 ~= null &
1000     sa..contents = sb..contents & sa..size = sb..size"
1001     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1002     ensures "True" */
1003 {
1004     Object r1a = sa.get(k1);
1005     sa.put(k2, v2);
1006     /*: assume "k1 ~= k2 | r1a = v2" */
1007     sb.put(k2, v2);
1008     Object r1b = sb.get(k1);
1009     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1010 }
1011
1012 static void get_put_post_c_32(HashTable sa, HashTable sb, Object k1, Object k2,
1013     Object v2)
1014 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1015     & k2 ~= null & v2 ~= null &
1016     sa..contents = sb..contents & sa..size = sb..size"
1017     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1018     ensures "True" */
1019 {
1020     Object r1a = sa.get(k1);
1021     sa.put(k2, v2);
1022     /*: assume "~(k1 ~= k2 | r1a = v2)" */
1023     sb.put(k2, v2);
1024     Object r1b = sb.get(k1);
1025     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1026 }
1027
1028 static void get_remove_pre_s_33(HashTable sa, HashTable sb, Object k1, Object k2)
1029 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1030     & k2 ~= null &
1031     sa..contents = sb..contents & sa..size = sb..size"
1032     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1033     ensures "True" */
1034 {
1035     /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
1036     Object r1a = sa.get(k1);
1037     Object r2a = sa.remove(k2);
1038     Object r2b = sb.remove(k2);
1039     Object r1b = sb.get(k1);
1040     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1041     sb..size" */
1042 }
1043
1044 static void get_remove_pre_c_33(HashTable sa, HashTable sb, Object k1, Object k2)
1045 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1046     & k2 ~= null &
1047     sa..contents = sb..contents & sa..size = sb..size"
1048     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1049     ensures "True" */
1050 {
1051     /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
1052     Object r1a = sa.get(k1);
1053     Object r2a = sa.remove(k2);

```

```

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 =
1058         sb..size)" */
1059 }
1060
1061 static void get_remove_between_s_34(HashTable sa, HashTable sb, Object k1, Object
1062     k2)
1063 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1064     & k2 ~= null &
1065         sa..contents = sb..contents & sa..size = sb..size"
1066     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1067     ensures "True" */
1068 {
1069     Object r1a = sa.get(k1);
1070     /*: assume "k1 ~= k2 | r1a = null" */
1071     Object r2a = sa.remove(k2);
1072
1073     Object r2b = sb.remove(k2);
1074     Object r1b = sb.get(k1);
1075
1076     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1077         sb..size" */
1078 }
1079
1080 static void get_remove_between_c_34(HashTable sa, HashTable sb, Object k1, Object
1081     k2)
1082 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1083     & k2 ~= null &
1084         sa..contents = sb..contents & sa..size = sb..size"
1085     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1086     ensures "True" */
1087 {
1088     Object r1a = sa.get(k1);
1089     /*: assume "~(k1 ~= k2 | r1a = null)" */
1090     Object r2a = sa.remove(k2);
1091
1092     Object r2b = sb.remove(k2);
1093     Object r1b = sb.get(k1);
1094
1095     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1096         sb..size)" */
1097 }
1098
1099 static void get_remove_post_s_35(HashTable sa, HashTable sb, Object k1, Object k2)
1100 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1101     & k2 ~= null &
1102         sa..contents = sb..contents & sa..size = sb..size"
1103     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1104     ensures "True" */
1105 {
1106     Object r1a = sa.get(k1);
1107     Object r2a = sa.remove(k2);
1108     /*: assume "k1 ~= k2 | r1a = null" */
1109
1110     Object r2b = sb.remove(k2);
1111     Object r1b = sb.get(k1);
1112
1113     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1114         sb..size" */
1115 }
1116
1117 static void get_remove_post_c_35(HashTable sa, HashTable sb, Object k1, Object k2)

```

```

1109  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1110      & k2 ~= null &
1111      sa..contents = sb..contents & sa..size = sb..size"
1112      modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1113      ensures "True" */
1114  {
1115      Object r1a = sa.get(k1);
1116      Object r2a = sa.remove(k2);
1117      /*: assume "~(k1 ~= k2 | r1a = null)" */
1118
1119      Object r2b = sb.remove(k2);
1120      Object r1b = sb.get(k1);
1121
1122      /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1123      sb..size)" */
1124  }
1125
1126  static void get_remove_pre_s_36(HashTable sa, HashTable sb, Object k1, Object k2)
1127  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1128      & k2 ~= null &
1129      sa..contents = sb..contents & sa..size = sb..size"
1130      modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1131      ensures "True" */
1132  {
1133      /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
1134      Object r1a = sa.get(k1);
1135      sa.remove(k2);
1136
1137      sb.remove(k2);
1138      Object r1b = sb.get(k1);
1139
1140      /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1141  }
1142
1143  static void get_remove_pre_c_36(HashTable sa, HashTable sb, Object k1, Object k2)
1144  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1145      & k2 ~= null &
1146      sa..contents = sb..contents & sa..size = sb..size"
1147      modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1148      ensures "True" */
1149  {
1150      /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
1151      Object r1a = sa.get(k1);
1152      sa.remove(k2);
1153
1154      sb.remove(k2);
1155      Object r1b = sb.get(k1);
1156
1157      /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1158  }
1159
1160  static void get_remove_between_s_37(HashTable sa, HashTable sb, Object k1, Object
1161      k2)
1162  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1163      & k2 ~= null &
1164      sa..contents = sb..contents & sa..size = sb..size"
1165      modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1166      ensures "True" */
1167  {
1168      Object r1a = sa.get(k1);
1169      /*: assume "k1 ~= k2 | r1a = null" */
1170      sa.remove(k2);
1171
1172      sb.remove(k2);
1173      Object r1b = sb.get(k1);

```

```

1168     /*: assert "ria = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1169 }
1170
1171 static void get_remove_between_c_37(HashTable sa, HashTable sb, Object k1, Object
1172     k2)
1173 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1174     & k2 ~= null &
1175     sa..contents = sb..contents & sa..size = sb..size"
1176     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1177     ensures "True" */
1178 {
1179     Object r1a = sa.get(k1);
1180     /*: assume "~(k1 ~= k2 | r1a = null)" */
1181     sa.remove(k2);
1182
1183     sb.remove(k2);
1184     Object r1b = sb.get(k1);
1185
1186     /*: assert "~(ria = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1187 }
1188
1189 static void get_remove_post_s_38(HashTable sa, HashTable sb, Object k1, Object k2)
1190 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1191     & k2 ~= null &
1192     sa..contents = sb..contents & sa..size = sb..size"
1193     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1194     ensures "True" */
1195 {
1196     Object r1a = sa.get(k1);
1197     sa.remove(k2);
1198     /*: assume "k1 ~= k2 | r1a = null" */
1199
1200     sb.remove(k2);
1201     Object r1b = sb.get(k1);
1202
1203     /*: assert "ria = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1204 }
1205
1206 static void get_remove_post_c_38(HashTable sa, HashTable sb, Object k1, Object k2)
1207 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1208     & k2 ~= null &
1209     sa..contents = sb..contents & sa..size = sb..size"
1210     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1211     ensures "True" */
1212 {
1213     Object r1a = sa.get(k1);
1214     sa.remove(k2);
1215     /*: assume "~(k1 ~= k2 | r1a = null)" */
1216
1217     sb.remove(k2);
1218     Object r1b = sb.get(k1);
1219
1220     /*: assert "~(ria = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1221 }
1222
1223 static void get_size_pre_s_39(HashTable sa, HashTable sb, Object k1)
1224 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1225     &
1226     sa..contents = sb..contents & sa..size = sb..size"
1227     ensures "True" */
1228 {
1229     /*: assume "True" */
1230     Object r1a = sa.get(k1);
1231     int r2a = sa.size();

```

```

1228     int r2b = sb.size();
1229     Object r1b = sb.get(k1);
1230
1231     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1232         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 =
1248         sb..size)" */
1249 }
1250
1251 static void get_size_between_s_40(HashTable sa, HashTable sb, Object k1)
1252 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
    &
1253         sa..contents = sb..contents & sa..size = sb..size"
1254     ensures "True" */
1255 {
1256     Object r1a = sa.get(k1);
1257     /*: assume "True" */
1258     int r2a = sa.size();
1259
1260     int r2b = sb.size();
1261     Object r1b = sb.get(k1);
1262
1263     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1264         sb..size" */
1265 }
1266
1267 static void get_size_between_c_40(HashTable sa, HashTable sb, Object k1)
1268 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
    &
1269         sa..contents = sb..contents & sa..size = sb..size"
1270     ensures "True" */
1271 {
1272     Object r1a = sa.get(k1);
1273     /*: assume "~(True)" */
1274     int r2a = sa.size();
1275
1276     int r2b = sb.size();
1277     Object r1b = sb.get(k1);
1278
1279     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1280         sb..size)" */
1281 }
1282
1283 static void get_size_post_s_41(HashTable sa, HashTable sb, Object k1)
1284 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
    &
1285         sa..contents = sb..contents & sa..size = sb..size"
1286     ensures "True" */
1287 {

```

```

1285     Object r1a = sa.get(k1);
1286     int r2a = sa.size();
1287     /*: assume "True" */
1288
1289     int r2b = sb.size();
1290     Object r1b = sb.get(k1);
1291
1292     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1293         sb..size" */
1294 }
1295
1296 static void get_size_post_c_41(HashTable sa, HashTable sb, Object k1)
1297 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1298     &
1299         sa..contents = sb..contents & sa..size = sb..size"
1300     ensures "True" */
1301 {
1302     Object r1a = sa.get(k1);
1303     int r2a = sa.size();
1304     /*: assume "~(True)" */
1305
1306     int r2b = sb.size();
1307     Object r1b = sb.get(k1);
1308
1309     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1310         sb..size)" */
1311 }
1312
1313 static void put_containsKey_pre_s_42(HashTable sa, HashTable sb, Object k1, Object
1314     v1, Object k2)
1315 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1316     & v1 ~= null & k2 ~= null &
1317         sa..contents = sb..contents & sa..size = sb..size"
1318     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1319     ensures "True" */
1320 {
1321     /*: assume "k1 ~= k2 | (EX v. (k1, v) : sa..contents)" */
1322     Object r1a = sa.put(k1, v1);
1323     boolean r2a = sa.containsKey(k2);
1324
1325     boolean r2b = sb.containsKey(k2);
1326     Object r1b = sb.put(k1, v1);
1327
1328     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1329         sb..size" */
1330 }
1331
1332 static void put_containsKey_pre_c_42(HashTable sa, HashTable sb, Object k1, Object
1333     v1, Object k2)
1334 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1335     & v1 ~= null & k2 ~= null &
1336         sa..contents = sb..contents & sa..size = sb..size"
1337     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1338     ensures "True" */
1339 {
1340     /*: assume "~(k1 ~= k2 | (EX v. (k1, v) : sa..contents))" */
1341     Object r1a = sa.put(k1, v1);
1342     boolean r2a = sa.containsKey(k2);
1343
1344     boolean r2b = sb.containsKey(k2);
1345     Object r1b = sb.put(k1, v1);
1346
1347     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1348         sb..size)" */
1349 }

```

```

1341 static void put_containsKey_between_s_43(HashTable sa, HashTable sb, Object k1,
1342     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" */

```

```

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 =
1404         sb..size)" */
1405 }
1406
1407 static void put_get_pre_s_45(HashTable sa, HashTable sb, Object k1, Object v1,
1408     Object k2)
1409 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1410     & v1 ~= null & k2 ~= null &
1411         sa..contents = sb..contents & sa..size = sb..size"
1412     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1413     ensures "True" */
1414 {
1415     /*: assume "k1 ~= k2 | (k1, v1) : sa..contents" */
1416     Object r1a = sa.put(k1, v1);
1417     Object r2a = sa.get(k2);
1418
1419     Object r2b = sb.get(k2);
1420     Object r1b = sb.put(k1, v1);
1421
1422     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1423         sb..size" */
1424 }
1425
1426 static void put_get_pre_c_45(HashTable sa, HashTable sb, Object k1, Object v1,
1427     Object k2)
1428 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1429     & v1 ~= null & k2 ~= null &
1430         sa..contents = sb..contents & sa..size = sb..size"
1431     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1432     ensures "True" */
1433 {
1434     /*: assume "~(k1 ~= k2 | (k1, v1) : sa..contents)" */
1435     Object r1a = sa.put(k1, v1);
1436     Object r2a = sa.get(k2);
1437
1438     Object r2b = sb.get(k2);
1439     Object r1b = sb.put(k1, v1);
1440
1441     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1442         sb..size)" */
1443 }
1444
1445 static void put_get_between_s_46(HashTable sa, HashTable sb, Object k1, Object v1,
1446     Object k2)
1447 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1448     & v1 ~= null & k2 ~= null &
1449         sa..contents = sb..contents & sa..size = sb..size"
1450     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1451     ensures "True" */
1452 {
1453     Object r1a = sa.put(k1, v1);
1454     /*: assume "k1 ~= k2 | r1a = v1" */
1455     Object r2a = sa.get(k2);
1456
1457     Object r2b = sb.get(k2);
1458     Object r1b = sb.put(k1, v1);
1459
1460 }

```



```

1451     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1452         sb..size" */
1453 }
1454 static void put_get_between_c_46(HashTable sa, HashTable sb, Object k1, Object v1,
1455     Object k2)
1456 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1457     & v1 ~= null & k2 ~= null &
1458         sa..contents = sb..contents & sa..size = sb..size"
1459     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1460     ensures "True" */
1461 {
1462     Object r1a = sa.put(k1, v1);
1463     /*: assume "~(k1 ~= k2 | r1a = v1)" */
1464     Object r2a = sa.get(k2);
1465
1466     Object r2b = sb.get(k2);
1467     Object r1b = sb.put(k1, v1);
1468
1469     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1470         sb..size)" */
1471 }
1472 static void put_get_post_s_47(HashTable sa, HashTable sb, Object k1, Object v1,
1473     Object k2)
1474 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1475     & v1 ~= null & k2 ~= null &
1476         sa..contents = sb..contents & sa..size = sb..size"
1477     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1478     ensures "True" */
1479 {
1480     Object r1a = sa.put(k1, v1);
1481     Object r2a = sa.get(k2);
1482     /*: assume "k1 ~= k2 | r1a = v1" */
1483
1484     Object r2b = sb.get(k2);
1485     Object r1b = sb.put(k1, v1);
1486
1487     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1488         sb..size" */
1489 }
1490 static void put_get_post_c_47(HashTable sa, HashTable sb, Object k1, Object v1,
1491     Object k2)
1492 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1493     & v1 ~= null & k2 ~= null &
1494         sa..contents = sb..contents & sa..size = sb..size"
1495     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1496     ensures "True" */
1497 {
1498     Object r1a = sa.put(k1, v1);
1499     Object r2a = sa.get(k2);
1500     /*: assume "~(k1 ~= k2 | r1a = v1)" */
1501
1502     Object r2b = sb.get(k2);
1503     Object r1b = sb.put(k1, v1);
1504
1505     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1506         sb..size)" */
1507 }
1508 static void put_put_pre_s_48(HashTable sa, HashTable sb, Object k1, Object v1,
1509     Object k2, Object v2)
1510 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1511     & 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 =
1516         sb..size" */
1517 }
1518
1519 static void put_put_pre_c_48(HashTable sa, HashTable sb, Object k1, Object v1,
1520     Object k2, Object v2)
1521 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1522     & v1 ~= null & k2 ~= null & v2 ~= null &
1523         sa..contents = sb..contents & sa..size = sb..size"
1524     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1525     ensures "True" */
1526 {
1527     /*: assume "~(k1 ~= k2 | ((k1, v1) : sa..contents & v1 = v2))" */
1528     Object r1a = sa.put(k1, v1);
1529     Object r2a = sa.put(k2, v2);
1530
1531     Object r2b = sb.put(k2, v2);
1532     Object r1b = sb.put(k1, v1);
1533
1534     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1535         sb..size)" */
1536 }
1537
1538 static void put_put_between_s_49(HashTable sa, HashTable sb, Object k1, Object v1,
1539     Object k2, Object v2)
1540 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1541     & v1 ~= null & k2 ~= null & v2 ~= null &
1542         sa..contents = sb..contents & sa..size = sb..size"
1543     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1544     ensures "True" */
1545 {
1546     Object r1a = sa.put(k1, v1);
1547     /*: assume "k1 ~= k2 | (r1a = v1 & v1 = v2)" */
1548     Object r2a = sa.put(k2, v2);
1549
1550     Object r2b = sb.put(k2, v2);
1551     Object r1b = sb.put(k1, v1);
1552
1553     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1554         sb..size" */
1555 }
1556
1557 static void put_put_between_c_49(HashTable sa, HashTable sb, Object k1, Object v1,
1558     Object k2, Object v2)
1559 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1560     & v1 ~= null & k2 ~= null & v2 ~= null &
1561         sa..contents = sb..contents & sa..size = sb..size"
1562     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1563     ensures "True" */
1564 {
1565     Object r1a = sa.put(k1, v1);
1566     /*: assume "~(k1 ~= k2 | (r1a = v1 & v1 = v2))" */
1567     Object r2a = sa.put(k2, v2);
1568
1569 }

```

```

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 =
1564         sb..size)" */
1565 }
1566
1567 static void put_put_post_s_50(HashTable sa, HashTable sb, Object k1, Object v1,
1568     Object k2, Object v2)
1569 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1570     & v1 ~= null & k2 ~= null & v2 ~= null &
1571         sa..contents = sb..contents & sa..size = sb..size"
1572     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1573     ensures "True" */
1574 {
1575     Object r1a = sa.put(k1, v1);
1576     Object r2a = sa.put(k2, v2);
1577     /*: assume "k1 ~= k2 | (r1a = v1 & v1 = v2)" */
1578
1579     Object r2b = sb.put(k2, v2);
1580     Object r1b = sb.put(k1, v1);
1581
1582     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1583         sb..size" */
1584 }
1585
1586 static void put_put_post_c_50(HashTable sa, HashTable sb, Object k1, Object v1,
1587     Object k2, Object v2)
1588 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1589     & v1 ~= null & k2 ~= null & v2 ~= null &
1590         sa..contents = sb..contents & sa..size = sb..size"
1591     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1592     ensures "True" */
1593 {
1594     Object r1a = sa.put(k1, v1);
1595     Object r2a = sa.put(k2, v2);
1596     /*: assume "~(k1 ~= k2 | (r1a = v1 & v1 = v2))" */
1597
1598     Object r2b = sb.put(k2, v2);
1599     Object r1b = sb.put(k1, v1);
1600
1601     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1602         sb..size)" */
1603 }
1604
1605 static void put_put_pre_s_51(HashTable sa, HashTable sb, Object k1, Object v1,
1606     Object k2, Object v2)
1607 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1608     & v1 ~= null & k2 ~= null & v2 ~= null &
1609         sa..contents = sb..contents & sa..size = sb..size"
1610     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1611     ensures "True" */
1612 {
1613     /*: assume "k1 ~= k2 | ((k1, v1) : sa..contents & v1 = v2)" */
1614     Object r1a = sa.put(k1, v1);
1615     sa.put(k2, v2);
1616
1617     sb.put(k2, v2);
1618     Object r1b = sb.put(k1, v1);
1619
1620     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1621 }
1622
1623 static void put_put_pre_c_51(HashTable sa, HashTable sb, Object k1, Object v1,
1624     Object k2, Object v2)

```

```

1615  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1616      & v1 ~= null & k2 ~= null & v2 ~= null &
1617          sa..contents = sb..contents & sa..size = sb..size"
1618      modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1619      ensures "True" */
1620  {
1621      /*: assume "~(k1 ~= k2 | ((k1, v1) : sa..contents & v1 = v2))" */
1622      Object r1a = sa.put(k1, v1);
1623      sa.put(k2, v2);
1624
1625      sb.put(k2, v2);
1626      Object r1b = sb.put(k1, v1);
1627
1628      /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1629  }
1630
1631  static void put_put_between_s_52(HashTable sa, HashTable sb, Object k1, Object v1,
1632      Object k2, Object v2)
1633  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1634      & v1 ~= null & k2 ~= null & v2 ~= null &
1635          sa..contents = sb..contents & sa..size = sb..size"
1636      modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1637      ensures "True" */
1638  {
1639      Object r1a = sa.put(k1, v1);
1640      /*: assume "k1 ~= k2 | (r1a = v1 & v1 = v2)" */
1641      sa.put(k2, v2);
1642
1643      sb.put(k2, v2);
1644      Object r1b = sb.put(k1, v1);
1645
1646      /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1647  }
1648
1649  static void put_put_between_c_52(HashTable sa, HashTable sb, Object k1, Object v1,
1650      Object k2, Object v2)
1651  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1652      & v1 ~= null & k2 ~= null & v2 ~= null &
1653          sa..contents = sb..contents & sa..size = sb..size"
1654      modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1655      ensures "True" */
1656  {
1657      Object r1a = sa.put(k1, v1);
1658      /*: assume "~(k1 ~= k2 | (r1a = v1 & v1 = v2))" */
1659      sa.put(k2, v2);
1660
1661      sb.put(k2, v2);
1662      Object r1b = sb.put(k1, v1);
1663
1664      /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1665  }
1666
1667  static void put_put_post_s_53(HashTable sa, HashTable sb, Object k1, Object v1,
1668      Object k2, Object v2)
1669  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1670      & v1 ~= null & k2 ~= null & v2 ~= null &
1671          sa..contents = sb..contents & sa..size = sb..size"
1672      modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1673      ensures "True" */
1674  {
1675      Object r1a = sa.put(k1, v1);
1676      sa.put(k2, v2);
1677      /*: assume "k1 ~= k2 | (r1a = v1 & v1 = v2)" */
1678
1679      sb.put(k2, v2);

```

```

1673     Object r1b = sb.put(k1, v1);
1674
1675     /*: assert "ria = 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,
1679     Object k2, Object v2)
1680 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1681 & v1 ~= null & k2 ~= null & v2 ~= null &
1682     sa..contents = sb..contents & sa..size = sb..size"
1683 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1684 ensures "True" */
1685 {
1686     Object r1a = sa.put(k1, v1);
1687     sa.put(k2, v2);
1688     /*: assume "~(k1 ~= k2 | (r1a = v1 & v1 = v2))" */
1689     sb.put(k2, v2);
1690     Object r1b = sb.put(k1, v1);
1691
1692     /*: assert "~(ria = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1693 }
1694
1695 static void put_remove_pre_s_54(HashTable sa, HashTable sb, Object k1, Object v1,
1696     Object k2)
1697 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1698 & v1 ~= null & k2 ~= null &
1699     sa..contents = sb..contents & sa..size = sb..size"
1700 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1701 ensures "True" */
1702 {
1703     /*: assume "k1 ~= k2" */
1704     Object r1a = sa.put(k1, v1);
1705     Object r2a = sa.remove(k2);
1706
1707     Object r2b = sb.remove(k2);
1708     Object r1b = sb.put(k1, v1);
1709
1710     /*: assert "ria = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1711     sb..size" */
1712 }
1713
1714 static void put_remove_pre_c_54(HashTable sa, HashTable sb, Object k1, Object v1,
1715     Object k2)
1716 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1717 & v1 ~= null & k2 ~= null &
1718     sa..contents = sb..contents & sa..size = sb..size"
1719 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1720 ensures "True" */
1721 {
1722     /*: assume "~(k1 ~= k2)" */
1723     Object r1a = sa.put(k1, v1);
1724     Object r2a = sa.remove(k2);
1725
1726     Object r2b = sb.remove(k2);
1727     Object r1b = sb.put(k1, v1);
1728
1729     /*: assert "~(ria = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1730     sb..size)" */
1731 }
1732
1733 static void put_remove_between_s_55(HashTable sa, HashTable sb, Object k1, Object
1734     v1, Object k2)
1735 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1736 & 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 =
1740         sb..size" */
1741 }
1742
1743 static void put_remove_between_c_55(HashTable sa, HashTable sb, Object k1, Object
1744     v1, Object k2)
1745 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1746     & v1 ~= null & k2 ~= null &
1747         sa..contents = sb..contents & sa..size = sb..size"
1748     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1749     ensures "True" */
1750 {
1751     Object r1a = sa.put(k1, v1);
1752     /*: assume "~(k1 ~= k2)" */
1753     Object r2a = sa.remove(k2);
1754
1755     Object r2b = sb.remove(k2);
1756     Object r1b = sb.put(k1, v1);
1757
1758     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1759         sb..size)" */
1760 }
1761
1762 static void put_remove_post_s_56(HashTable sa, HashTable sb, Object k1, Object v1,
1763     Object k2)
1764 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1765     & v1 ~= null & k2 ~= null &
1766         sa..contents = sb..contents & sa..size = sb..size"
1767     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1768     ensures "True" */
1769 {
1770     Object r1a = sa.put(k1, v1);
1771     Object r2a = sa.remove(k2);
1772     /*: assume "k1 ~= k2" */
1773
1774     Object r2b = sb.remove(k2);
1775     Object r1b = sb.put(k1, v1);
1776
1777     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1778         sb..size" */
1779 }
1780
1781 static void put_remove_post_c_56(HashTable sa, HashTable sb, Object k1, Object v1,
1782     Object k2)
1783 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1784     & v1 ~= null & k2 ~= null &
1785         sa..contents = sb..contents & sa..size = sb..size"
1786     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1787     ensures "True" */
1788 {
1789     Object r1a = sa.put(k1, v1);
1790     Object r2a = sa.remove(k2);
1791     /*: assume "~(k1 ~= k2)" */
1792
1793 }

```

```

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 =
1788         sb..size)" */
1789 }
1790
1791 static void put_remove_pre_s_57(HashTable sa, HashTable sb, Object k1, Object v1,
1792     Object k2)
1793 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1794     & v1 ~= null & k2 ~= null &
1795         sa..contents = sb..contents & sa..size = sb..size"
1796     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1797     ensures "True" */
1798 {
1799     /*: assume "k1 ~= k2" */
1800     Object r1a = sa.put(k1, v1);
1801     sa.remove(k2);
1802
1803     sb.remove(k2);
1804     Object r1b = sb.put(k1, v1);
1805
1806     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1807 }
1808
1809 static void put_remove_pre_c_57(HashTable sa, HashTable sb, Object k1, Object v1,
1810     Object k2)
1811 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1812     & v1 ~= null & k2 ~= null &
1813         sa..contents = sb..contents & sa..size = sb..size"
1814     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1815     ensures "True" */
1816 {
1817     /*: assume "~(k1 ~= k2)" */
1818     Object r1a = sa.put(k1, v1);
1819     sa.remove(k2);
1820
1821     sb.remove(k2);
1822     Object r1b = sb.put(k1, v1);
1823
1824     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1825 }
1826
1827 static void put_remove_between_s_58(HashTable sa, HashTable sb, Object k1, Object
1828     v1, Object k2)
1829 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1830     & v1 ~= null & k2 ~= null &
1831         sa..contents = sb..contents & sa..size = sb..size"
1832     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1833     ensures "True" */
1834 {
1835     Object r1a = sa.put(k1, v1);
1836     /*: assume "k1 ~= k2" */
1837     sa.remove(k2);
1838
1839     sb.remove(k2);
1840     Object r1b = sb.put(k1, v1);
1841
1842     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1843 }
1844
1845 static void put_remove_between_c_58(HashTable sa, HashTable sb, Object k1, Object
1846     v1, Object k2)
1847 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1848     & 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,
1855     Object k2)
1856 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1857     & v1 ~= null & k2 ~= null &
1858     sa..contents = sb..contents & sa..size = sb..size"
1859     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1860     ensures "True" */
1861 {
1862     Object r1a = sa.put(k1, v1);
1863     sa.remove(k2);
1864     /*: assume "k1 ~= k2" */
1865
1866     sb.remove(k2);
1867     Object r1b = sb.put(k1, v1);
1868
1869     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
1870 }
1871
1872 static void put_remove_post_c_59(HashTable sa, HashTable sb, Object k1, Object v1,
1873     Object k2)
1874 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1875     & v1 ~= null & k2 ~= null &
1876     sa..contents = sb..contents & sa..size = sb..size"
1877     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1878     ensures "True" */
1879 {
1880     Object r1a = sa.put(k1, v1);
1881     sa.remove(k2);
1882     /*: assume "~(k1 ~= k2)" */
1883
1884     sb.remove(k2);
1885     Object r1b = sb.put(k1, v1);
1886
1887     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
1888 }
1889
1890 static void put_size_pre_s_60(HashTable sa, HashTable sb, Object k1, Object v1)
1891 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1892     & v1 ~= null &
1893     sa..contents = sb..contents & sa..size = sb..size"
1894     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1895     ensures "True" */
1896 {
1897     /*: assume "EX v. (k1, v) : sa..contents" */
1898     Object r1a = sa.put(k1, v1);
1899     int r2a = sa.size();
1900
1901     int r2b = sb.size();
1902     Object r1b = sb.put(k1, v1);

```



```

1899     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1900         sb..size" */
1901 }
1902
1903 static void put_size_pre_c_60(HashTable sa, HashTable sb, Object k1, Object v1)
1904 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1905     & v1 ~= null &
1906         sa..contents = sb..contents & sa..size = sb..size"
1907     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1908     ensures "True" */
1909 {
1910     /*: assume "~(EX v. (k1, v) : sa..contents)" */
1911     Object r1a = sa.put(k1, v1);
1912     int r2a = sa.size();
1913
1914     int r2b = sb.size();
1915     Object r1b = sb.put(k1, v1);
1916
1917     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1918         sb..size)" */
1919 }
1920
1921 static void put_size_between_s_61(HashTable sa, HashTable sb, Object k1, Object v1)
1922 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1923     & v1 ~= null &
1924         sa..contents = sb..contents & sa..size = sb..size"
1925     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1926     ensures "True" */
1927 {
1928     Object r1a = sa.put(k1, v1);
1929     /*: assume "r1a ~= null" */
1930     int r2a = sa.size();
1931
1932     int r2b = sb.size();
1933     Object r1b = sb.put(k1, v1);
1934
1935     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1936         sb..size" */
1937 }
1938
1939 static void put_size_between_c_61(HashTable sa, HashTable sb, Object k1, Object v1)
1940 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1941     & v1 ~= null &
1942         sa..contents = sb..contents & sa..size = sb..size"
1943     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1944     ensures "True" */
1945 {
1946     Object r1a = sa.put(k1, v1);
1947     /*: assume "~(r1a ~= null)" */
1948     int r2a = sa.size();
1949
1950     int r2b = sb.size();
1951     Object r1b = sb.put(k1, v1);
1952
1953     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1954         sb..size)" */
1955 }
1956
1957 static void put_size_post_s_62(HashTable sa, HashTable sb, Object k1, Object v1)
1958 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1959     & 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 {

```

```

1956     Object r1a = sa.put(k1, v1);
1957     int r2a = sa.size();
1958     /*: assume "r1a ~= null" */
1959
1960     int r2b = sb.size();
1961     Object r1b = sb.put(k1, v1);
1962
1963     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1964         sb..size" */
1965 }
1966
1967 static void put_size_post_c_62(HashTable sa, HashTable sb, Object k1, Object v1)
1968 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1969     & v1 ~= null &
1970         sa..contents = sb..contents & sa..size = sb..size"
1971     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1972     ensures "True" */
1973 {
1974     Object r1a = sa.put(k1, v1);
1975     int r2a = sa.size();
1976     /*: assume "~(r1a ~= null)" */
1977
1978     int r2b = sb.size();
1979     Object r1b = sb.put(k1, v1);
1980
1981     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
1982         sb..size)" */
1983 }
1984
1985 static void put_containsKey_pre_s_63(HashTable sa, HashTable sb, Object k1, Object
1986     v1, Object k2)
1987 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
1988     & v1 ~= null & k2 ~= null &
1989         sa..contents = sb..contents & sa..size = sb..size"
1990     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
1991     ensures "True" */
1992 {
1993     /*: assume "k1 ~= k2 | (EX v. (k1, v) : sa..contents)" */
1994     sa.put(k1, v1);
1995     boolean r2a = sa.containsKey(k2);
1996
1997     boolean r2b = sb.containsKey(k2);
1998     sb.put(k1, v1);
1999
2000     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2001 }
2002
2003 static void put_containsKey_pre_c_63(HashTable sa, HashTable sb, Object k1, Object
2004     v1, Object k2)
2005 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2006     & v1 ~= null & k2 ~= null &
2007         sa..contents = sb..contents & sa..size = sb..size"
2008     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2009     ensures "True" */
2010 {
2011     /*: assume "~(k1 ~= k2 | (EX v. (k1, v) : sa..contents))" */
2012     sa.put(k1, v1);
2013     boolean r2a = sa.containsKey(k2);
2014
2015     boolean r2b = sb.containsKey(k2);
2016     sb.put(k1, v1);
2017
2018     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2019 }

```

```

2014 static void put_containsKey_between_s_64(HashTable sa, HashTable sb, Object k1,
2015     Object v1, Object k2)
2016 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2017     & v1 ~= null & k2 ~= null &
2018     sa..contents = sb..contents & sa..size = sb..size"
2019     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2020     ensures "True" */
2021 {
2022     sa.put(k1, v1);
2023     /*: assume "k1 ~= k2 | (EX v. (k1, v) : sa..(old contents))" */
2024     boolean r2a = sa.containsKey(k2);
2025
2026     boolean r2b = sb.containsKey(k2);
2027     sb.put(k1, v1);
2028
2029     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2030 }
2031
2032 static void put_containsKey_between_c_64(HashTable sa, HashTable sb, Object k1,
2033     Object v1, Object k2)
2034 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2035     & v1 ~= null & k2 ~= null &
2036     sa..contents = sb..contents & sa..size = sb..size"
2037     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2038     ensures "True" */
2039 {
2040     sa.put(k1, v1);
2041     /*: assume "~(k1 ~= k2 | (EX v. (k1, v) : sa..(old contents)))" */
2042     boolean r2a = sa.containsKey(k2);
2043
2044     boolean r2b = sb.containsKey(k2);
2045     sb.put(k1, v1);
2046
2047     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2048 }
2049
2050 static void put_containsKey_post_s_65(HashTable sa, HashTable sb, Object k1, Object
2051     v1, Object k2)
2052 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2053     & v1 ~= null & k2 ~= null &
2054     sa..contents = sb..contents & sa..size = sb..size"
2055     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2056     ensures "True" */
2057 {
2058     sa.put(k1, v1);
2059     boolean r2a = sa.containsKey(k2);
2060     /*: assume "k1 ~= k2 | (EX v. (k1, v) : sa..(old contents))" */
2061
2062     boolean r2b = sb.containsKey(k2);
2063     sb.put(k1, v1);
2064
2065     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2066 }
2067
2068 static void put_containsKey_post_c_65(HashTable sa, HashTable sb, Object k1, Object
2069     v1, Object k2)
2070 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2071     & v1 ~= null & k2 ~= null &
2072     sa..contents = sb..contents & sa..size = sb..size"
2073     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2074     ensures "True" */
2075 {
2076     sa.put(k1, v1);
2077     boolean r2a = sa.containsKey(k2);
2078     /*: assume "~(k1 ~= k2 | (EX v. (k1, v) : sa..(old contents)))" */

```

```

2071     boolean r2b = sb.containsKey(k2);
2072     sb.put(k1, v1);
2073
2074     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2075 }
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,
2143     Object k2)
2144 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2145     & v1 ~= null & k2 ~= null &
2146     sa..contents = sb..contents & sa..size = sb..size"
2147     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2148     ensures "True" */
2149 {
2150     sa.put(k1, v1);
2151     Object r2a = sa.get(k2);
2152     /*: assume "k1 ~= k2 | (k1, v1) : sa..(old contents)" */
2153
2154     Object r2b = sb.get(k2);
2155     sb.put(k1, v1);
2156
2157     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2158 }
2159
2160 static void put_get_post_c_68(HashTable sa, HashTable sb, Object k1, Object v1,
2161     Object k2)
2162 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2163     & v1 ~= null & k2 ~= null &
2164     sa..contents = sb..contents & sa..size = sb..size"
2165     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2166     ensures "True" */
2167 {
2168     sa.put(k1, v1);
2169     Object r2a = sa.get(k2);
2170     /*: assume "~(k1 ~= k2 | (k1, v1) : sa..(old contents))" */
2171
2172     Object r2b = sb.get(k2);
2173     sb.put(k1, v1);
2174
2175     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2176 }
2177
2178 static void put_put_pre_s_69(HashTable sa, HashTable sb, Object k1, Object v1,
2179     Object k2, Object v2)
2180 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2181     & v1 ~= null & k2 ~= null & v2 ~= null &
2182     sa..contents = sb..contents & sa..size = sb..size"
2183     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2184     ensures "True" */
2185 {
2186     /*: assume "k1 ~= k2 | ((k1, v1) : sa..contents & v1 = v2)" */
2187     sa.put(k1, v1);
2188     Object r2a = sa.put(k2, v2);
2189
2190     Object r2b = sb.put(k2, v2);
2191     sb.put(k1, v1);
2192 }

```

```

2187     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2188 }
2189
2190 static void put_put_pre_c_69(HashTable sa, HashTable sb, Object k1, Object v1,
2191     Object k2, Object v2)
2192 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2193     & v1 ~= null & k2 ~= null & v2 ~= null &
2194     sa..contents = sb..contents & sa..size = sb..size"
2195     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2196     ensures "True" */
2197 {
2198     /*: assume "~(k1 ~= k2 | ((k1, v1) : sa..contents & v1 = v2))" */
2199     sa.put(k1, v1);
2200     Object r2a = sa.put(k2, v2);
2201
2202     Object r2b = sb.put(k2, v2);
2203     sb.put(k1, v1);
2204
2205     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2206 }
2207
2208 static void put_put_between_s_70(HashTable sa, HashTable sb, Object k1, Object v1,
2209     Object k2, Object v2)
2210 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2211     & v1 ~= null & k2 ~= null & v2 ~= null &
2212     sa..contents = sb..contents & sa..size = sb..size"
2213     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2214     ensures "True" */
2215 {
2216     sa.put(k1, v1);
2217     /*: assume "k1 ~= k2 | ((k1, v1) : sa..(old contents) & v1 = v2)" */
2218     Object r2a = sa.put(k2, v2);
2219
2220     Object r2b = sb.put(k2, v2);
2221     sb.put(k1, v1);
2222
2223     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2224 }
2225
2226 static void put_put_between_c_70(HashTable sa, HashTable sb, Object k1, Object v1,
2227     Object k2, Object v2)
2228 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2229     & v1 ~= null & k2 ~= null & v2 ~= null &
2230     sa..contents = sb..contents & sa..size = sb..size"
2231     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2232     ensures "True" */
2233 {
2234     sa.put(k1, v1);
2235     /*: assume "~(k1 ~= k2 | ((k1, v1) : sa..(old contents) & v1 = v2))" */
2236     Object r2a = sa.put(k2, v2);
2237
2238     Object r2b = sb.put(k2, v2);
2239     sb.put(k1, v1);
2240
2241     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2242 }
2243
2244 static void put_put_post_s_71(HashTable sa, HashTable sb, Object k1, Object v1,
2245     Object k2, Object v2)
2246 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2247     & v1 ~= null & k2 ~= null & v2 ~= null &
2248     sa..contents = sb..contents & sa..size = sb..size"
2249     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2250     ensures "True" */
2251 {

```

```

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,
2255     Object k2, Object v2)
2256 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2257     & v1 ~= null & k2 ~= null & v2 ~= null &
2258     sa..contents = sb..contents & sa..size = sb..size"
2259     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2260     ensures "True" */
2261 {
2262     sa.put(k1, v1);
2263     Object r2a = sa.put(k2, v2);
2264     /*: assume "~(k1 ~= k2 | ((k1, v1) : sa..(old contents) & v1 = v2))" */
2265
2266     Object r2b = sb.put(k2, v2);
2267     sb.put(k1, v1);
2268
2269     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2270 }
2271
2272 static void put_put_pre_s_72(HashTable sa, HashTable sb, Object k1, Object v1,
2273     Object k2, Object v2)
2274 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2275     & v1 ~= null & k2 ~= null & v2 ~= null &
2276     sa..contents = sb..contents & sa..size = sb..size"
2277     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2278     ensures "True" */
2279 {
2280     /*: assume "k1 ~= k2 | v1 = v2" */
2281     sa.put(k1, v1);
2282     sa.put(k2, v2);
2283
2284     sb.put(k2, v2);
2285     sb.put(k1, v1);
2286
2287     /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
2288 }
2289
2290 static void put_put_pre_c_72(HashTable sa, HashTable sb, Object k1, Object v1,
2291     Object k2, Object v2)
2292 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2293     & v1 ~= null & k2 ~= null & v2 ~= null &
2294     sa..contents = sb..contents & sa..size = sb..size"
2295     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2296     ensures "True" */
2297 {
2298     /*: assume "~(k1 ~= k2 | v1 = v2)" */
2299     sa.put(k1, v1);
2300     sa.put(k2, v2);
2301
2302     sb.put(k2, v2);
2303     sb.put(k1, v1);
2304
2305     /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
2306 }

```

```

2302 static void put_put_between_s_73(HashTable sa, HashTable sb, Object k1, Object v1,
2303     Object k2, Object v2)
2304 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2305     & v1 ~= null & k2 ~= null & v2 ~= null &
2306         sa..contents = sb..contents & sa..size = sb..size"
2307     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2308     ensures "True" */
2309 {
2310     sa.put(k1, v1);
2311     /*: assume "k1 ~= k2 | v1 = v2" */
2312     sa.put(k2, v2);
2313
2314     sb.put(k2, v2);
2315     sb.put(k1, v1);
2316
2317     /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
2318 }
2319
2320 static void put_put_between_c_73(HashTable sa, HashTable sb, Object k1, Object v1,
2321     Object k2, Object v2)
2322 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2323     & v1 ~= null & k2 ~= null & v2 ~= null &
2324         sa..contents = sb..contents & sa..size = sb..size"
2325     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2326     ensures "True" */
2327 {
2328     sa.put(k1, v1);
2329     /*: assume "~(k1 ~= k2 | v1 = v2)" */
2330     sa.put(k2, v2);
2331
2332     sb.put(k2, v2);
2333     sb.put(k1, v1);
2334
2335     /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
2336 }
2337
2338 static void put_put_post_s_74(HashTable sa, HashTable sb, Object k1, Object v1,
2339     Object k2, Object v2)
2340 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2341     & v1 ~= null & k2 ~= null & v2 ~= null &
2342         sa..contents = sb..contents & sa..size = sb..size"
2343     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2344     ensures "True" */
2345 {
2346     sa.put(k1, v1);
2347     sa.put(k2, v2);
2348     /*: assume "k1 ~= k2 | v1 = v2" */
2349
2350     sb.put(k2, v2);
2351     sb.put(k1, v1);
2352
2353     /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
2354 }
2355
2356 static void put_put_post_c_74(HashTable sa, HashTable sb, Object k1, Object v1,
2357     Object k2, Object v2)
2358 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2359     & v1 ~= null & k2 ~= 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.put(k1, v1);
2365     sa.put(k2, v2);
2366     /*: assume "~(k1 ~= k2 | v1 = v2)" */

```



```

2359     sb.put(k2, v2);
2360     sb.put(k1, v1);
2361
2362     /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
2363 }
2364
2365 static void put_remove_pre_s_75(HashTable sa, HashTable sb, Object k1, Object v1,
2366     Object k2)
2367 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2368     & v1 ~= null & k2 ~= null &
2369         sa..contents = sb..contents & sa..size = sb..size"
2370     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2371     ensures "True" */
2372 {
2373     /*: assume "k1 ~= k2" */
2374     sa.put(k1, v1);
2375     Object r2a = sa.remove(k2);
2376
2377     Object r2b = sb.remove(k2);
2378     sb.put(k1, v1);
2379
2380     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2381 }
2382
2383 static void put_remove_pre_c_75(HashTable sa, HashTable sb, Object k1, Object v1,
2384     Object k2)
2385 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2386     & v1 ~= null & k2 ~= null &
2387         sa..contents = sb..contents & sa..size = sb..size"
2388     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2389     ensures "True" */
2390 {
2391     /*: assume "~(k1 ~= k2)" */
2392     sa.put(k1, v1);
2393     Object r2a = sa.remove(k2);
2394
2395     Object r2b = sb.remove(k2);
2396     sb.put(k1, v1);
2397
2398     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2399 }
2400
2401 static void put_remove_between_s_76(HashTable sa, HashTable sb, Object k1, Object
2402     v1, Object k2)
2403 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2404     & v1 ~= null & k2 ~= null &
2405         sa..contents = sb..contents & sa..size = sb..size"
2406     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2407     ensures "True" */
2408 {
2409     sa.put(k1, v1);
2410     /*: assume "k1 ~= k2" */
2411     Object r2a = sa.remove(k2);
2412
2413     Object r2b = sb.remove(k2);
2414     sb.put(k1, v1);
2415
2416     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2417 }
2418
2419 static void put_remove_between_c_76(HashTable sa, HashTable sb, Object k1, Object
2420     v1, Object k2)
2421 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2422     & 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,
2431     Object k2)
2432 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2433     & v1 ~= null & k2 ~= null &
2434     sa..contents = sb..contents & sa..size = sb..size"
2435     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2436     ensures "True" */
2437 {
2438     sa.put(k1, v1);
2439     Object r2a = sa.remove(k2);
2440     /*: assume "k1 ~= k2" */
2441
2442     Object r2b = sb.remove(k2);
2443     sb.put(k1, v1);
2444
2445     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2446 }
2447
2448 static void put_remove_post_c_77(HashTable sa, HashTable sb, Object k1, Object v1,
2449     Object k2)
2450 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2451     & v1 ~= null & k2 ~= null &
2452     sa..contents = sb..contents & sa..size = sb..size"
2453     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2454     ensures "True" */
2455 {
2456     sa.put(k1, v1);
2457     Object r2a = sa.remove(k2);
2458     /*: assume "~(k1 ~= k2)" */
2459
2460     Object r2b = sb.remove(k2);
2461     sb.put(k1, v1);
2462
2463     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2464 }
2465
2466 static void put_remove_pre_s_78(HashTable sa, HashTable sb, Object k1, Object v1,
2467     Object k2)
2468 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2469     & v1 ~= null & k2 ~= null &
2470     sa..contents = sb..contents & sa..size = sb..size"
2471     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2472     ensures "True" */
2473 {
2474     /*: assume "k1 ~= k2" */
2475     sa.put(k1, v1);
2476     sa.remove(k2);
2477
2478     sb.remove(k2);
2479     sb.put(k1, v1);
2480 }

```

```

2475     /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
2476 }
2477
2478 static void put_remove_pre_c_78(HashTable sa, HashTable sb, Object k1, Object v1,
2479     Object k2)
2479 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2480     & v1 ~= null & k2 ~= null &
2481         sa..contents = sb..contents & sa..size = sb..size"
2482     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2483     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
2494 static void put_remove_between_s_79(HashTable sa, HashTable sb, Object k1, Object
2495     v1, Object k2)
2495 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2496     & v1 ~= null & k2 ~= null &
2497         sa..contents = sb..contents & sa..size = sb..size"
2498     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2499     ensures "True" */
2499 {
2500     sa.put(k1, v1);
2501     /*: assume "k1 ~= k2" */
2502     sa.remove(k2);
2503
2504     sb.remove(k2);
2505     sb.put(k1, v1);
2506
2507     /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
2508 }
2509
2510 static void put_remove_between_c_79(HashTable sa, HashTable sb, Object k1, Object
2511     v1, Object k2)
2511 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2512     & v1 ~= null & k2 ~= null &
2513         sa..contents = sb..contents & sa..size = sb..size"
2514     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2515     ensures "True" */
2515 {
2516     sa.put(k1, v1);
2517     /*: assume "~(k1 ~= k2)" */
2518     sa.remove(k2);
2519
2520     sb.remove(k2);
2521     sb.put(k1, v1);
2522
2523     /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
2524 }
2525
2526 static void put_remove_post_s_80(HashTable sa, HashTable sb, Object k1, Object v1,
2527     Object k2)
2527 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2528     & v1 ~= null & k2 ~= null &
2529         sa..contents = sb..contents & sa..size = sb..size"
2530     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2531     ensures "True" */
2531 {

```

```

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,
2543     Object k2)
2544 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2545     & v1 ~= null & k2 ~= null &
2546     sa..contents = sb..contents & sa..size = sb..size"
2547     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2548     ensures "True" */
2549 {
2550     sa.put(k1, v1);
2551     sa.remove(k2);
2552     /*: assume "~(k1 ~= k2)" */
2553
2554     sb.remove(k2);
2555     sb.put(k1, v1);
2556
2557     /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
2558 }
2559
2560 static void put_size_pre_s_81(HashTable sa, HashTable sb, Object k1, Object v1)
2561 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2562     & v1 ~= null &
2563     sa..contents = sb..contents & sa..size = sb..size"
2564     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2565     ensures "True" */
2566 {
2567     /*: assume "EX v. (k1, v) : sa..contents" */
2568     sa.put(k1, v1);
2569     int r2a = sa.size();
2570
2571     int r2b = sb.size();
2572     sb.put(k1, v1);
2573
2574     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
2575 }
2576
2577 static void put_size_pre_c_81(HashTable sa, HashTable sb, Object k1, Object v1)
2578 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2579     & v1 ~= null &
2580     sa..contents = sb..contents & sa..size = sb..size"
2581     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2582     ensures "True" */
2583 {
2584     /*: assume "~(EX v. (k1, v) : sa..contents)" */
2585     sa.put(k1, v1);
2586     int r2a = sa.size();
2587
2588     int r2b = sb.size();
2589     sb.put(k1, v1);
2590
2591     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
2592 }
2593
2594 static void put_size_between_s_82(HashTable sa, HashTable sb, Object k1, Object v1)
2595 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2596     & 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(HashTable sa, HashTable sb, Object k1, Object v1)
2607 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashTable sa, HashTable sb, Object k1, Object v1)
2623 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashTable sa, HashTable sb, Object k1, Object v1)
2639 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashTable sa, HashTable sb, Object k1,
2655     Object k2)
2656 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2657     & k2 ~= null &
2658     sa..contents = sb..contents & sa..size = sb..size"
2659     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2660     ensures "True" */
2661 {
2662     /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
2663     Object r1a = sa.remove(k1);
2664     boolean r2a = sa.containsKey(k2);
2665
2666     boolean r2b = sb.containsKey(k2);
2667     Object r1b = sb.remove(k1);
2668
2669     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2670     sb..size" */
2671 }
2672
2673 static void remove_containsKey_pre_c_84(HashTable sa, HashTable sb, Object k1,
2674     Object k2)
2675 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2676     & k2 ~= null &
2677     sa..contents = sb..contents & sa..size = sb..size"
2678     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2679     ensures "True" */
2680 {
2681     /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
2682     Object r1a = sa.remove(k1);
2683     boolean r2a = sa.containsKey(k2);
2684
2685     boolean r2b = sb.containsKey(k2);
2686     Object r1b = sb.remove(k1);
2687
2688     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2689     sb..size)" */
2690 }
2691
2692 static void remove_containsKey_between_s_85(HashTable sa, HashTable sb, Object k1,
2693     Object k2)
2694 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2695     & k2 ~= 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     Object r1a = sa.remove(k1);
2701     /*: assume "k1 ~= k2 | r1a = null" */
2702     boolean r2a = sa.containsKey(k2);
2703
2704     boolean r2b = sb.containsKey(k2);
2705     Object r1b = sb.remove(k1);
2706
2707     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2708     sb..size" */
2709 }
2710
2711 static void remove_containsKey_between_c_85(HashTable sa, HashTable sb, Object k1,
2712     Object k2)
2713 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2714     & k2 ~= null &
2715     sa..contents = sb..contents & sa..size = sb..size"
2716     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2717     ensures "True" */
2718 {

```

```

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 =
2716         sb..size)" */
2717 }
2718
2719 static void remove_containsKey_post_s_86(HashTable sa, HashTable sb, Object k1,
2720     Object k2)
2721 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2722     & k2 ~= null &
2723         sa..contents = sb..contents & sa..size = sb..size"
2724     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2725     ensures "True" */
2726 {
2727     Object r1a = sa.remove(k1);
2728     boolean r2a = sa.containsKey(k2);
2729     /*: assume "k1 ~= k2 | r1a = null" */
2730
2731     boolean r2b = sb.containsKey(k2);
2732     Object r1b = sb.remove(k1);
2733
2734     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2735         sb..size" */
2736 }
2737
2738 static void remove_containsKey_post_c_86(HashTable sa, HashTable sb, Object k1,
2739     Object k2)
2740 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2741     & k2 ~= null &
2742         sa..contents = sb..contents & sa..size = sb..size"
2743     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2744     ensures "True" */
2745 {
2746     Object r1a = sa.remove(k1);
2747     boolean r2a = sa.containsKey(k2);
2748     /*: assume "~(k1 ~= k2 | r1a = null)" */
2749
2750     boolean r2b = sb.containsKey(k2);
2751     Object r1b = sb.remove(k1);
2752
2753     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2754         sb..size)" */
2755 }
2756
2757 static void remove_get_pre_s_87(HashTable sa, HashTable sb, Object k1, Object k2)
2758 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2759     & k2 ~= null &
2760         sa..contents = sb..contents & sa..size = sb..size"
2761     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2762     ensures "True" */
2763 {
2764     /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
2765     Object r1a = sa.remove(k1);
2766     Object r2a = sa.get(k2);
2767
2768     Object r2b = sb.get(k2);
2769     Object r1b = sb.remove(k1);
2770
2771     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2772         sb..size" */

```

```

2764 }
2765
2766 static void remove_get_pre_c_87(HashTable sa, HashTable sb, Object k1, Object k2)
2767 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashTable sa, HashTable sb, Object k1, Object
      k2)
2783 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashTable sa, HashTable sb, Object k1, Object
      k2)
2799 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashTable sa, HashTable sb, Object k1, Object k2)
2815 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(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
2879 k2, Object v2)
2880 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2881 & k2 ~= null & v2 ~= null &
2882 sa..contents = sb..contents & sa..size = sb..size"
2883 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2884 ensures "True" */
2885 {
2886 Object r1a = sa.remove(k1);
2887 /*: assume "k1 ~= k2" */
2888 Object r2a = sa.put(k2, v2);
2889
2890 Object r2b = sb.put(k2, v2);
2891 Object r1b = sb.remove(k1);
2892
2893 /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2894 sb..size" */
2895 }
2896
2897 static void remove_put_between_c_91(HashTable sa, HashTable sb, Object k1, Object
2898 k2, Object v2)
2899 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2900 & k2 ~= null & v2 ~= null &
2901 sa..contents = sb..contents & sa..size = sb..size"
2902 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2903 ensures "True" */
2904 {
2905 Object r1a = sa.remove(k1);
2906 /*: assume "~(k1 ~= k2)" */
2907 Object r2a = sa.put(k2, v2);
2908
2909 Object r2b = sb.put(k2, v2);
2910 Object r1b = sb.remove(k1);
2911
2912 /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2913 sb..size)" */
2914 }
2915
2916 static void remove_put_post_s_92(HashTable sa, HashTable sb, Object k1, Object k2,
2917 Object v2)
2918 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2919 & k2 ~= null & 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 Object r1a = sa.remove(k1);
2925 Object r2a = sa.put(k2, v2);
2926 /*: assume "k1 ~= k2" */
2927
2928 Object r2b = sb.put(k2, v2);
2929 Object r1b = sb.remove(k1);
2930
2931 /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
2932 sb..size" */
2933 }
2934
2935 static void remove_put_post_c_92(HashTable sa, HashTable sb, Object k1, Object k2,
2936 Object v2)
2937 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2938 & k2 ~= null & v2 ~= null &
2939 sa..contents = sb..contents & sa..size = sb..size"
2940 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
2991 k2, Object v2)
2992 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
2993 & k2 ~= null & v2 ~= null &
2994 sa..contents = sb..contents & sa..size = sb..size"
2995 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
2996 ensures "True" */
2997 {
2998 Object r1a = sa.remove(k1);
2999 /*: assume "~(k1 ~= k2)" */
3000 sa.put(k2, v2);
3001
3002 sb.put(k2, v2);
3003 Object r1b = sb.remove(k1);
3004
3005 /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
3006 }
3007
3008 static void remove_put_post_s_95(HashTable sa, HashTable sb, Object k1, Object k2,
3009 Object v2)
3010 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3011 & k2 ~= null & v2 ~= null &
3012 sa..contents = sb..contents & sa..size = sb..size"
3013 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3014 ensures "True" */
3015 {
3016 Object r1a = sa.remove(k1);
3017 sa.put(k2, v2);
3018 /*: assume "k1 ~= k2" */
3019
3020 sb.put(k2, v2);
3021 Object r1b = sb.remove(k1);
3022
3023 /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
3024 }
3025
3026 static void remove_put_post_c_95(HashTable sa, HashTable sb, Object k1, Object k2,
3027 Object v2)
3028 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3029 & k2 ~= null & v2 ~= null &
3030 sa..contents = sb..contents & sa..size = sb..size"
3031 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3032 ensures "True" */
3033 {
3034 Object r1a = sa.remove(k1);
3035 sa.put(k2, v2);
3036 /*: assume "~(k1 ~= k2)" */
3037
3038 sb.put(k2, v2);
3039 Object r1b = sb.remove(k1);
3040
3041 /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
3042 }
3043
3044 static void remove_remove_pre_s_96(HashTable sa, HashTable sb, Object k1, Object k2)
3045 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3046 & k2 ~= null &
3047 sa..contents = sb..contents & sa..size = sb..size"
3048 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3049 ensures "True" */
3050 {
3051 /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
3052 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(HashTable sa, HashTable sb, Object k1, Object k2)
3055 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashTable sa, HashTable sb, Object k1,
        Object k2)
3071 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
3072         sa..contents = sb..contents & sa..size = sb..size"
3073     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3074     ensures "True" */
3075 {
3076     Object r1a = sa.remove(k1);
3077     /*: assume "k1 ~= k2 | r1a = null" */
3078     Object r2a = sa.remove(k2);
3079
3080     Object r2b = sb.remove(k2);
3081     Object r1b = sb.remove(k1);
3082
3083     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
        sb..size" */
3084 }
3085
3086 static void remove_remove_between_c_97(HashTable sa, HashTable sb, Object k1,
        Object k2)
3087 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
        & k2 ~= null &
3088         sa..contents = sb..contents & sa..size = sb..size"
3089     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3090     ensures "True" */
3091 {
3092     Object r1a = sa.remove(k1);
3093     /*: assume "~(k1 ~= k2 | r1a = null)" */
3094     Object r2a = sa.remove(k2);
3095
3096     Object r2b = sb.remove(k2);
3097     Object r1b = sb.remove(k1);
3098
3099     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
        sb..size)" */
3100 }
3101

```

```

3102 static void remove_remove_post_s_98(HashTable sa, HashTable sb, Object k1, Object
3103 k2)
3104 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3105 & k2 ~= null &
3106         sa..contents = sb..contents & sa..size = sb..size"
3107     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3108     ensures "True" */
3109 {
3110     Object r1a = sa.remove(k1);
3111     Object r2a = sa.remove(k2);
3112     /*: assume "k1 ~= k2 | r1a = null" */
3113     Object r2b = sb.remove(k2);
3114     Object r1b = sb.remove(k1);
3115     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
3116         sb..size" */
3117 }
3118 static void remove_remove_post_c_98(HashTable sa, HashTable sb, Object k1, Object
3119 k2)
3120 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3121 & k2 ~= null &
3122         sa..contents = sb..contents & sa..size = sb..size"
3123     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3124     ensures "True" */
3125 {
3126     Object r1a = sa.remove(k1);
3127     Object r2a = sa.remove(k2);
3128     /*: assume "~(k1 ~= k2 | r1a = null)" */
3129     Object r2b = sb.remove(k2);
3130     Object r1b = sb.remove(k1);
3131     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
3132         sb..size)" */
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
3136 & k2 ~= null &
3137         sa..contents = sb..contents & sa..size = sb..size"
3138     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3139     ensures "True" */
3140 {
3141     /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
3142     Object r1a = sa.remove(k1);
3143     sa.remove(k2);
3144     sb.remove(k2);
3145     Object r1b = sb.remove(k1);
3146     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..size = sb..size" */
3147 }
3148 static void remove_remove_pre_c_99(HashTable sa, HashTable sb, Object k1, Object k2)
3149 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3150 & k2 ~= null &
3151         sa..contents = sb..contents & sa..size = sb..size"
3152     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3153     ensures "True" */
3154 {
3155     /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
3156     Object r1a = sa.remove(k1);
3157     sa.remove(k2);
3158 }

```

```

3159     sb.remove(k2);
3160     Object r1b = sb.remove(k1);
3161
3162     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..size = sb..size)" */
3163 }
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

```



```

3275     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
3276         sb..size" */
3277 }
3278 static void remove_size_between_c_103(HashTable sa, HashTable sb, Object k1)
3279 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3280     &
3281         sa..contents = sb..contents & sa..size = sb..size"
3282     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3283     ensures "True" */
3284 {
3285     Object r1a = sa.remove(k1);
3286     /*: assume "~(r1a = null)" */
3287     int r2a = sa.size();
3288
3289     int r2b = sb.size();
3290     Object r1b = sb.remove(k1);
3291
3292     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
3293         sb..size)" */
3294 }
3295 static void remove_size_post_s_104(HashTable sa, HashTable sb, Object k1)
3296 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3297     &
3298         sa..contents = sb..contents & sa..size = sb..size"
3299     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3300     ensures "True" */
3301 {
3302     Object r1a = sa.remove(k1);
3303     int r2a = sa.size();
3304     /*: assume "r1a = null" */
3305
3306     int r2b = sb.size();
3307     Object r1b = sb.remove(k1);
3308
3309     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
3310         sb..size" */
3311 }
3312 static void remove_size_post_c_104(HashTable sa, HashTable sb, Object k1)
3313 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3314     &
3315         sa..contents = sb..contents & sa..size = sb..size"
3316     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3317     ensures "True" */
3318 {
3319     Object r1a = sa.remove(k1);
3320     int r2a = sa.size();
3321     /*: assume "~(r1a = null)" */
3322
3323     int r2b = sb.size();
3324     Object r1b = sb.remove(k1);
3325
3326     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
3327         sb..size)" */
3328 }
3329 static void remove_containsKey_pre_s_105(HashTable sa, HashTable sb, Object k1,
3330     Object k2)
3331 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3332     & k2 ~= null &
3333         sa..contents = sb..contents & sa..size = sb..size"
3334     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3335     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,
3343     Object k2)
3344 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3345     & k2 ~= null &
3346     sa..contents = sb..contents & sa..size = sb..size"
3347     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3348     ensures "True" */
3349 {
3350     /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
3351     sa.remove(k1);
3352     boolean r2a = sa.containsKey(k2);
3353
3354     boolean r2b = sb.containsKey(k2);
3355     sb.remove(k1);
3356
3357     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3358 }
3359
3360 static void remove_containsKey_between_s_106(HashTable sa, HashTable sb, Object k1,
3361     Object k2)
3362 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3363     & k2 ~= null &
3364     sa..contents = sb..contents & sa..size = sb..size"
3365     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3366     ensures "True" */
3367 {
3368     sa.remove(k1);
3369     /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents))" */
3370     boolean r2a = sa.containsKey(k2);
3371
3372     boolean r2b = sb.containsKey(k2);
3373     sb.remove(k1);
3374
3375     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3376 }
3377
3378 static void remove_containsKey_between_c_106(HashTable sa, HashTable sb, Object k1,
3379     Object k2)
3380 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3381     & k2 ~= null &
3382     sa..contents = sb..contents & sa..size = sb..size"
3383     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3384     ensures "True" */
3385 {
3386     sa.remove(k1);
3387     /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents)))" */
3388     boolean r2a = sa.containsKey(k2);
3389
3390     boolean r2b = sb.containsKey(k2);
3391     sb.remove(k1);
3392
3393     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3394 }

```

```

3390 static void remove_containsKey_post_s_107(HashTable sa, HashTable sb, Object k1,
3391 Object k2)
3392 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3393 & k2 ~= null &
3394 sa..contents = sb..contents & sa..size = sb..size"
3395 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3396 ensures "True" */
3397 {
3398 sa.remove(k1);
3399 boolean r2a = sa.containsKey(k2);
3400 /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents))" */
3401
3402 boolean r2b = sb.containsKey(k2);
3403 sb.remove(k1);
3404
3405 /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3406 }
3407
3408 static void remove_containsKey_post_c_107(HashTable sa, HashTable sb, Object k1,
3409 Object k2)
3410 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3411 & k2 ~= null &
3412 sa..contents = sb..contents & sa..size = sb..size"
3413 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3414 ensures "True" */
3415 {
3416 sa.remove(k1);
3417 boolean r2a = sa.containsKey(k2);
3418 /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents)))" */
3419
3420 boolean r2b = sb.containsKey(k2);
3421 sb.remove(k1);
3422
3423 /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3424 }
3425
3426 static void remove_get_pre_s_108(HashTable sa, HashTable sb, Object k1, Object k2)
3427 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3428 & k2 ~= null &
3429 sa..contents = sb..contents & sa..size = sb..size"
3430 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3431 ensures "True" */
3432 {
3433 /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
3434 sa.remove(k1);
3435 Object r2a = sa.get(k2);
3436
3437 Object r2b = sb.get(k2);
3438 sb.remove(k1);
3439
3440 /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3441 }
3442
3443 static void remove_get_pre_c_108(HashTable sa, HashTable sb, Object k1, Object k2)
3444 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3445 & k2 ~= null &
3446 sa..contents = sb..contents & sa..size = sb..size"
3447 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3448 ensures "True" */
3449 {
3450 /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
3451 sa.remove(k1);
3452 Object r2a = sa.get(k2);
3453
3454 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
3455     k2)
3456 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3457     & k2 ~= null &
3458     sa..contents = sb..contents & sa..size = sb..size"
3459     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3460     ensures "True" */
3461 {
3462     sa.remove(k1);
3463     /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents))" */
3464     Object r2a = sa.get(k2);
3465
3466     Object r2b = sb.get(k2);
3467     sb.remove(k1);
3468
3469     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3470 }
3471
3472 static void remove_get_between_c_109(HashTable sa, HashTable sb, Object k1, Object
3473     k2)
3474 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3475     & k2 ~= null &
3476     sa..contents = sb..contents & sa..size = sb..size"
3477     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3478     ensures "True" */
3479 {
3480     sa.remove(k1);
3481     /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents)))" */
3482     Object r2a = sa.get(k2);
3483
3484     Object r2b = sb.get(k2);
3485     sb.remove(k1);
3486
3487     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3488 }
3489
3490 static void remove_get_post_s_110(HashTable sa, HashTable sb, Object k1, Object k2)
3491 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3492     & k2 ~= null &
3493     sa..contents = sb..contents & sa..size = sb..size"
3494     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3495     ensures "True" */
3496 {
3497     sa.remove(k1);
3498     Object r2a = sa.get(k2);
3499     /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents))" */
3500
3501     Object r2b = sb.get(k2);
3502     sb.remove(k1);
3503
3504     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3505 }
3506
3507 static void remove_get_post_c_110(HashTable sa, HashTable sb, Object k1, Object k2)
3508 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3509     & k2 ~= null &
3510     sa..contents = sb..contents & sa..size = sb..size"
3511     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3512     ensures "True" */
3513 {

```

```

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,
3519     Object v2)
3520 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3521     & k2 ~= null & v2 ~= null &
3522     sa..contents = sb..contents & sa..size = sb..size"
3523     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3524     ensures "True" */
3525 {
3526     /*: assume "k1 ~= k2" */
3527     sa.remove(k1);
3528     Object r2a = sa.put(k2, v2);
3529
3530     Object r2b = sb.put(k2, v2);
3531     sb.remove(k1);
3532
3533     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3534 }
3535
3536 static void remove_put_pre_c_111(HashTable sa, HashTable sb, Object k1, Object k2,
3537     Object v2)
3538 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3539     & k2 ~= null & v2 ~= null &
3540     sa..contents = sb..contents & sa..size = sb..size"
3541     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3542     ensures "True" */
3543 {
3544     /*: assume "~(k1 ~= k2)" */
3545     sa.remove(k1);
3546     Object r2a = sa.put(k2, v2);
3547
3548     Object r2b = sb.put(k2, v2);
3549     sb.remove(k1);
3550
3551     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3552 }
3553
3554 static void remove_put_between_s_112(HashTable sa, HashTable sb, Object k1, Object
3555     k2, Object v2)
3556 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3557     & k2 ~= null & v2 ~= null &
3558     sa..contents = sb..contents & sa..size = sb..size"
3559     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3560     ensures "True" */
3561 {
3562     sa.remove(k1);
3563     /*: assume "k1 ~= k2" */
3564     Object r2a = sa.put(k2, v2);
3565
3566     Object r2b = sb.put(k2, v2);
3567     sb.remove(k1);
3568
3569     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3570 }

```

```

3566 static void remove_put_between_c_112(HashTable sa, HashTable sb, Object k1, Object
3567 k2, Object v2)
3568 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3569 & k2 ~= null & v2 ~= null &
3570     sa..contents = sb..contents & sa..size = sb..size"
3571 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3572 ensures "True" */
3573 {
3574     sa.remove(k1);
3575     /*: assume "~(k1 ~= k2)" */
3576     Object r2a = sa.put(k2, v2);
3577
3578     Object r2b = sb.put(k2, v2);
3579     sb.remove(k1);
3580
3581     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3582 }
3583
3584 static void remove_put_post_s_113(HashTable sa, HashTable sb, Object k1, Object k2,
3585 Object v2)
3586 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3587 & k2 ~= null & v2 ~= null &
3588     sa..contents = sb..contents & sa..size = sb..size"
3589 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3590 ensures "True" */
3591 {
3592     sa.remove(k1);
3593     Object r2a = sa.put(k2, v2);
3594     /*: assume "k1 ~= k2" */
3595
3596     Object r2b = sb.put(k2, v2);
3597     sb.remove(k1);
3598
3599     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3600 }
3601
3602 static void remove_put_post_c_113(HashTable sa, HashTable sb, Object k1, Object k2,
3603 Object v2)
3604 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3605 & k2 ~= null & v2 ~= null &
3606     sa..contents = sb..contents & sa..size = sb..size"
3607 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3608 ensures "True" */
3609 {
3610     sa.remove(k1);
3611     Object r2a = sa.put(k2, v2);
3612     /*: assume "~(k1 ~= k2)" */
3613
3614     Object r2b = sb.put(k2, v2);
3615     sb.remove(k1);
3616
3617     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3618 }
3619
3620 static void remove_put_pre_s_114(HashTable sa, HashTable sb, Object k1, Object k2,
3621 Object v2)
3622 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3623 & k2 ~= null & v2 ~= null &
3624     sa..contents = sb..contents & sa..size = sb..size"
3625 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3626 ensures "True" */
3627 {
3628     /*: assume "k1 ~= k2" */
3629     sa.remove(k1);
3630     sa.put(k2, v2);

```

```

3623     sb.put(k2, v2);
3624     sb.remove(k1);
3625
3626     /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
3627 }
3628
3629
3630 static void remove_put_pre_c_114(HashTable sa, HashTable sb, Object k1, Object k2,
3631     Object v2)
3632 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3633     & k2 ~= null & v2 ~= null &
3634     sa..contents = sb..contents & sa..size = sb..size"
3635     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3636     ensures "True" */
3637 {
3638     /*: assume "~(k1 ~= k2)" */
3639     sa.remove(k1);
3640     sa.put(k2, v2);
3641
3642     sb.put(k2, v2);
3643     sb.remove(k1);
3644
3645     /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
3646 }
3647
3648 static void remove_put_between_s_115(HashTable sa, HashTable sb, Object k1, Object
3649     k2, Object v2)
3650 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3651     & k2 ~= null & v2 ~= null &
3652     sa..contents = sb..contents & sa..size = sb..size"
3653     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3654     ensures "True" */
3655 {
3656     sa.remove(k1);
3657     /*: assume "k1 ~= k2" */
3658     sa.put(k2, v2);
3659
3660     sb.put(k2, v2);
3661     sb.remove(k1);
3662
3663     /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
3664 }
3665
3666 static void remove_put_between_c_115(HashTable sa, HashTable sb, Object k1, Object
3667     k2, Object v2)
3668 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3669     & k2 ~= null & v2 ~= null &
3670     sa..contents = sb..contents & sa..size = sb..size"
3671     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3672     ensures "True" */
3673 {
3674     sa.remove(k1);
3675     /*: assume "~(k1 ~= k2)" */
3676     sa.put(k2, v2);
3677
3678     sb.put(k2, v2);
3679     sb.remove(k1);
3680
3681     /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
3682 }
3683
3684 static void remove_put_post_s_116(HashTable sa, HashTable sb, Object k1, Object k2,
3685     Object v2)
3686 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3687     & 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,
3695     Object v2)
3696 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3697     & k2 ~= null & v2 ~= null &
3698     sa..contents = sb..contents & sa..size = sb..size"
3699     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3700     ensures "True" */
3701 {
3702     sa.remove(k1);
3703     sa.put(k2, v2);
3704     /*: assume "~(k1 ~= k2)" */
3705
3706     sb.put(k2, v2);
3707     sb.remove(k1);
3708
3709     /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
3710 }
3711
3712 static void remove_remove_pre_s_117(HashTable sa, HashTable sb, Object k1, Object
3713     k2)
3714 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3715     & k2 ~= null &
3716     sa..contents = sb..contents & sa..size = sb..size"
3717     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3718     ensures "True" */
3719 {
3720     /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..contents)" */
3721     sa.remove(k1);
3722     Object r2a = sa.remove(k2);
3723
3724     Object r2b = sb.remove(k2);
3725     sb.remove(k1);
3726
3727     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3728 }
3729
3730 static void remove_remove_pre_c_117(HashTable sa, HashTable sb, Object k1, Object
3731     k2)
3732 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3733     & k2 ~= null &
3734     sa..contents = sb..contents & sa..size = sb..size"
3735     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3736     ensures "True" */
3737 {
3738     /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..contents))" */
3739     sa.remove(k1);
3740     Object r2a = sa.remove(k2);
3741
3742     Object r2b = sb.remove(k2);
3743     sb.remove(k1);
3744 }

```



```

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,
3743     Object k2)
3744 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3745     & k2 ~= null &
3746     sa..contents = sb..contents & sa..size = sb..size"
3747     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3748     ensures "True" */
3749 {
3750     sa.remove(k1);
3751     /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents))" */
3752     Object r2a = sa.remove(k2);
3753
3754     Object r2b = sb.remove(k2);
3755     sb.remove(k1);
3756
3757     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3758 }
3759
3760 static void remove_remove_between_c_118(HashTable sa, HashTable sb, Object k1,
3761     Object k2)
3762 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3763     & k2 ~= null &
3764     sa..contents = sb..contents & sa..size = sb..size"
3765     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3766     ensures "True" */
3767 {
3768     sa.remove(k1);
3769     /*: assume "~(k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents)))" */
3770     Object r2a = sa.remove(k2);
3771
3772     Object r2b = sb.remove(k2);
3773     sb.remove(k1);
3774
3775     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3776 }
3777
3778 static void remove_remove_post_s_119(HashTable sa, HashTable sb, Object k1, Object
3779     k2)
3780 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3781     & k2 ~= null &
3782     sa..contents = sb..contents & sa..size = sb..size"
3783     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3784     ensures "True" */
3785 {
3786     sa.remove(k1);
3787     Object r2a = sa.remove(k2);
3788     /*: assume "k1 ~= k2 | ~(EX v. (k1, v) : sa..(old contents))" */
3789
3790     Object r2b = sb.remove(k2);
3791     sb.remove(k1);
3792
3793     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3794 }
3795
3796 static void remove_remove_post_c_119(HashTable sa, HashTable sb, Object k1, Object
3797     k2)
3798 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3799     & k2 ~= null &
3800     sa..contents = sb..contents & sa..size = sb..size"
3801     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3802     ensures "True" */
3803 {

```

```

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
3807     k2)
3808 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3809     & k2 ~= null &
3810     sa..contents = sb..contents & sa..size = sb..size"
3811     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3812     ensures "True" */
3813 {
3814     /*: assume "True" */
3815     sa.remove(k1);
3816     sa.remove(k2);
3817
3818     sb.remove(k2);
3819     sb.remove(k1);
3820
3821     /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
3822 }
3823
3824 static void remove_remove_pre_c_120(HashTable sa, HashTable sb, Object k1, Object
3825     k2)
3826 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3827     & k2 ~= null &
3828     sa..contents = sb..contents & sa..size = sb..size"
3829     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3830     ensures "True" */
3831 {
3832     /*: assume "~(True)" */
3833     sa.remove(k1);
3834     sa.remove(k2);
3835
3836     sb.remove(k2);
3837     sb.remove(k1);
3838
3839     /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
3840 }
3841
3842 static void remove_remove_between_s_121(HashTable sa, HashTable sb, Object k1,
3843     Object k2)
3844 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3845     & k2 ~= null &
3846     sa..contents = sb..contents & sa..size = sb..size"
3847     modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3848     ensures "True" */
3849 {
3850     sa.remove(k1);
3851     /*: assume "True" */
3852     sa.remove(k2);
3853
3854     sb.remove(k2);
3855     sb.remove(k1);
3856
3857     /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
3858 }

```

```

3854 static void remove_remove_between_c_121(HashTable sa, HashTable sb, Object k1,
3855 Object k2)
3856 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3857 & k2 ~= null &
3858     sa..contents = sb..contents & sa..size = sb..size"
3859 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3860 ensures "True" */
3861 {
3862     sa.remove(k1);
3863     /*: assume "~(True)" */
3864     sa.remove(k2);
3865
3866     sb.remove(k2);
3867     sb.remove(k1);
3868
3869     /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
3870 }
3871
3872 static void remove_remove_post_s_122(HashTable sa, HashTable sb, Object k1, Object
3873 k2)
3874 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3875 & k2 ~= null &
3876     sa..contents = sb..contents & sa..size = sb..size"
3877 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3878 ensures "True" */
3879 {
3880     sa.remove(k1);
3881     sa.remove(k2);
3882     /*: assume "True" */
3883
3884     sb.remove(k2);
3885     sb.remove(k1);
3886
3887     /*: assert "sa..contents = sb..contents & sa..size = sb..size" */
3888 }
3889
3890 static void remove_remove_post_c_122(HashTable sa, HashTable sb, Object k1, Object
3891 k2)
3892 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3893 & k2 ~= null &
3894     sa..contents = sb..contents & sa..size = sb..size"
3895 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3896 ensures "True" */
3897 {
3898     sa.remove(k1);
3899     sa.remove(k2);
3900     /*: assume "~(True)" */
3901
3902     sb.remove(k2);
3903     sb.remove(k1);
3904
3905     /*: assert "~(sa..contents = sb..contents & sa..size = sb..size)" */
3906 }
3907
3908 static void remove_size_pre_s_123(HashTable sa, HashTable sb, Object k1)
3909 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
3910 &
3911     sa..contents = sb..contents & sa..size = sb..size"
3912 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3913 ensures "True" */
3914 {
3915     /*: assume "~(EX v. (k1, v) : sa..contents)" */
3916     sa.remove(k1);
3917     int r2a = sa.size();

```

```

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(HashTable sa, HashTable sb, Object k1)
3919 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashTable sa, HashTable sb, Object k1)
3935 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashTable sa, HashTable sb, Object k1)
3951 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashTable sa, HashTable sb, Object k1)
3967 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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 {
3972     sa.remove(k1);

```

```

3973     int r2a = sa.size();
3974     /*: assume "~(EX v. (k1, v) : sa..(old contents))" */
3975
3976     int r2b = sb.size();
3977     sb.remove(k1);
3978
3979     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..size = sb..size" */
3980 }
3981
3982 static void remove_size_post_c_125(HashTable sa, HashTable sb, Object k1)
3983 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k1 ~= null
    &
3984         sa..contents = sb..contents & sa..size = sb..size"
3985 modifies "sa..contents", "sb..contents", "sa..size", "sb..size"
3986 ensures "True" */
3987 {
3988     sa.remove(k1);
3989     int r2a = sa.size();
3990     /*: assume "~(~(EX v. (k1, v) : sa..(old contents)))" */
3991
3992     int r2b = sb.size();
3993     sb.remove(k1);
3994
3995     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..size = sb..size)" */
3996 }
3997
3998 static void size_containsKey_pre_s_126(HashTable sa, HashTable sb, Object k2)
3999 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
    &
4000         sa..contents = sb..contents & sa..size = sb..size"
4001 ensures "True" */
4002 {
4003     /*: assume "True" */
4004     int r1a = sa.size();
4005     boolean r2a = sa.containsKey(k2);
4006
4007     boolean r2b = sb.containsKey(k2);
4008     int r1b = sb.size();
4009
4010     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
        sb..size" */
4011 }
4012
4013 static void size_containsKey_pre_c_126(HashTable sa, HashTable sb, Object k2)
4014 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
    &
4015         sa..contents = sb..contents & sa..size = sb..size"
4016 ensures "True" */
4017 {
4018     /*: assume "~(True)" */
4019     int r1a = sa.size();
4020     boolean r2a = sa.containsKey(k2);
4021
4022     boolean r2b = sb.containsKey(k2);
4023     int r1b = sb.size();
4024
4025     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
        sb..size)" */
4026 }
4027
4028 static void size_containsKey_between_s_127(HashTable sa, HashTable sb, Object k2)
4029 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
    &
4030         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(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)

```

```

4089  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
      &
4090          sa..contents = sb..contents & sa..size = sb..size"
4091  ensures "True" */
4092  {
4093    /*: assume "True" */
4094    int r1a = sa.size();
4095    Object r2a = sa.get(k2);
4096
4097    Object r2b = sb.get(k2);
4098    int r1b = sb.size();
4099
4100    /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
      sb..size" */
4101  }
4102
4103  static void size_get_pre_c_129(HashTable sa, HashTable sb, Object k2)
4104  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
      &
4105          sa..contents = sb..contents & sa..size = sb..size"
4106  ensures "True" */
4107  {
4108    /*: assume "~(True)" */
4109    int r1a = sa.size();
4110    Object r2a = sa.get(k2);
4111
4112    Object r2b = sb.get(k2);
4113    int r1b = sb.size();
4114
4115    /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
      sb..size)" */
4116  }
4117
4118  static void size_get_between_s_130(HashTable sa, HashTable sb, Object k2)
4119  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
      &
4120          sa..contents = sb..contents & sa..size = sb..size"
4121  ensures "True" */
4122  {
4123    int r1a = sa.size();
4124    /*: assume "True" */
4125    Object r2a = sa.get(k2);
4126
4127    Object r2b = sb.get(k2);
4128    int r1b = sb.size();
4129
4130    /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
      sb..size" */
4131  }
4132
4133  static void size_get_between_c_130(HashTable sa, HashTable sb, Object k2)
4134  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & k2 ~= null
      &
4135          sa..contents = sb..contents & sa..size = sb..size"
4136  ensures "True" */
4137  {
4138    int r1a = sa.size();
4139    /*: assume "~(True)" */
4140    Object r2a = sa.get(k2);
4141
4142    Object r2b = sb.get(k2);
4143    int r1b = sb.size();
4144
4145    /*: 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

```



```

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(HashTable sa, HashTable sb, Object k2, Object v2)
4211 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashTable sa, HashTable sb, Object k2, Object v2)
4227 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashTable sa, HashTable sb, Object k2, Object v2)
4243 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashTable sa, HashTable sb, Object k2, Object v2)
4259 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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 {
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);

```

```

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(HashTable sa, HashTable sb, Object k2)
4389 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashTable sa, HashTable sb, Object k2)
4405 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashTable sa, HashTable sb, Object k2)
4421 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashTable sa, HashTable sb, Object k2)
4437 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashTable sa, HashTable sb, Object k2)
4453 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashTable sa, HashTable sb, Object k2)
4469 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(HashTable sa, HashTable sb, Object k2)
4485 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init & 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(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);

```

```

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(HashTable sa, HashTable sb)
4567 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
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 =
4579     sb..size" */
4580 }
4581
4582 static void size_size_pre_c_144(HashTable sa, HashTable sb)
4583 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4584     sa..contents = sb..contents & sa..size = sb..size"
4585     ensures "True" */
4586 {
4587     /*: assume "~(True)" */
4588     int r1a = sa.size();
4589     int r2a = sa.size();
4590
4591     int r2b = sb.size();
4592     int r1b = sb.size();
4593
4594     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
4595     sb..size)" */
4596 }
4597
4598 static void size_size_between_s_145(HashTable sa, HashTable sb)
4599 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4600     sa..contents = sb..contents & sa..size = sb..size"
4601     ensures "True" */
4602 {
4603     int r1a = sa.size();
4604     /*: assume "True" */
4605     int r2a = sa.size();
4606
4607     int r2b = sb.size();
4608     int r1b = sb.size();
4609
4610     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
4611     sb..size" */
4612 }
4613
4614 static void size_size_between_c_145(HashTable sa, HashTable sb)
4615 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4616     sa..contents = sb..contents & sa..size = sb..size"
4617     ensures "True" */
4618 {
4619     int r1a = sa.size();
4620     /*: assume "~(True)" */
4621     int r2a = sa.size();
4622
4623     int r2b = sb.size();
4624     int r1b = sb.size();

```

```

4623     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
4624         sb..size)" */
4625 }
4626 static void size_size_post_s_146(HashTable sa, HashTable sb)
4627 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4628     sa..contents = sb..contents & sa..size = sb..size"
4629     ensures "True" */
4630 {
4631     int r1a = sa.size();
4632     int r2a = sa.size();
4633     /*: assume "True" */
4634
4635     int r2b = sb.size();
4636     int r1b = sb.size();
4637
4638     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
4639         sb..size" */
4640 }
4641 static void size_size_post_c_146(HashTable sa, HashTable sb)
4642 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4643     sa..contents = sb..contents & sa..size = sb..size"
4644     ensures "True" */
4645 {
4646     int r1a = sa.size();
4647     int r2a = sa.size();
4648     /*: assume "~(True)" */
4649
4650     int r2b = sb.size();
4651     int r1b = sb.size();
4652
4653     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..size =
4654         sb..size)" */
4655 }
4656 }

```

### B.5.3 Inverse Testing Methods

Listing 15. HashTableInv.java

```

1 class HashTableInv {
2     static void put_0(HashTable s, Object k, Object v)
3     /*: requires "s ~= null & s..init & k ~= null & v ~= null"
4         modifies "s..contents", "s..size"
5         ensures "True" */
6     {
7         Object r = s.put(k, v);
8         if (r != null) { s.put(k, r); } else { s.remove(k); }
9
10        /*: assert "s..contents = s..(old contents) & s..size = s..(old size)" */
11    }
12
13    static void remove_1(HashTable s, Object k)
14    /*: requires "s ~= null & s..init & k ~= null"
15        modifies "s..contents", "s..size"
16        ensures "True" */
17    {
18        Object r = s.remove(k);
19        if (r != null) { s.put(k, r); }
20
21        /*: assert "s..contents = s..(old contents) & s..size = s..(old size)" */
22    }
23 }

```



24 }

## B.6 ArrayList

### B.6.1 Data Structure

Listing 16. ArrayList.java

```
1 public class ArrayList
2 {
3     private Object[] elementData;
4     private int size;
5
6     /*: public specvar init :: bool;
7         vardefs "init == elementData ~= null"
8
9         invariant InitInv: "~init --> size = 0 & elementData = null"
10        invariant ElementDataInv: "init --> elementData : hidden"
11        invariant NotHiddenInv: "init --> (ALL i. ((0 <= i & i < size) -->
12            elementData.[i] ~: hidden))"
13        invariant InjInv: "ALL x y. x ~= y & x..elementData ~= null --> x..elementData
14            ~= y..elementData";
15        invariant SizeInv: "init --> 0 <= size & size <= elementData..Array.length"
16
17        public specvar contents :: "(int * obj) set";
18        vardefs "contents == {(i, n). 0 <= i & i < size & n = elementData.[i]}"
19        public specvar csize :: int;
20        vardefs "csize == size"
21        public specvar msize :: int;
22        vardefs "msize == elementData..Array.length"
23
24        public ensured invariant "init --> (ALL i v. (i, v) : contents --> 0 <= i & i <
25            csize)"
26        public ensured invariant RangeInvInj: "init --> (ALL i v1 v2. (i, v1) :
27            contents & (i, v2) : contents --> v1 = v2)" */
28
29        public ArrayList(int initialCapacity)
30        /*: requires "~init & 0 < initialCapacity"
31            modifies "init", "msize"
32            ensures "init & contents = {} & csize = 0 & msize = initialCapacity" */
33        {
34            elementData = new /*: hidden */ Object[initialCapacity];
35        }
36
37        private void _ensureCapacity(int minCapacity)
38        /*: requires "init & theinvs"
39            modifies "Array.arrayState", "elementData", "msize", "ArrayList.hidden"
40            ensures "minCapacity <= msize & old msize <= msize & (ALL x. x..init =
41                (fieldRead (old ArrayList.init) x)) & (ALL a j. a ~= elementData --> a.[j] =
42                arrayRead (old arrayState) a j) & (ALL j. ((0 <= j & j < size) -->
43                elementData.[j] = arrayRead (old arrayState) (old elementData) j)) & (ALL
44                o1. (o1 : old alloc) --> ((o1 : old hidden) = (o1 : hidden))) & theinvs" */
45        {
46            int oldCapacity = elementData.length;
47            if (minCapacity > oldCapacity) {
48                Object oldData[] = elementData;
49                int newCapacity = (oldCapacity * 3)/2 + 1;
50                if (newCapacity < minCapacity)
51                    newCapacity = minCapacity;
52                elementData = new /*: hidden */ Object[newCapacity];
53                int i = 0;
54                while /*: invariant "0 <= i & (ALL j. ((0 <= j & j < size --> oldData.[j] =
55                    arrayRead (old arrayState) (old elementData) j))) & (ALL j. ((0 <= j & j
56                    < i) --> elementData.[j] = oldData.[j])) & (ALL a j. a ~= elementData
57                    --> a.[j] = arrayRead (old arrayState) a j)" */ (i < size) {
58                    elementData[i] = oldData[i];
59                }
60            }
61        }
62    }
63 }
```

```

48         i = i + 1;
49     }
50 }
51 }
52
53 public int size()
54 /*: requires "init"
55    ensures "result = csize" */
56 {
57     return size;
58 }
59
60 public int indexOf(Object elem)
61 /*: requires "init"
62    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)" */
63 {
64     return indexOfInt(elem);
65 }
66
67 private int indexOfInt(Object elem)
68 /*: requires "init & theinvs"
69    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" */
70 {
71     int i = 0;
72
73     while /*: invariant "0 <= i & (ALL j. 0 <= j & j < i --> elem ~=
        elementData.[j])" */ (i < size) {
74         if (elementData[i] == elem)
75             return i;
76         i = i + 1;
77     }
78     return -1;
79 }
80
81 public int lastIndexOf(Object elem)
82 /*: requires "init"
83    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)" */
84 {
85     int i = size - 1;
86     while /*: invariant "i < size & (ALL j. i < j & j < size --> elem ~=
        elementData.[j])" */ (i >= 0) {
87         if (elementData[i] == elem)
88             return i;
89         i = i - 1;
90     }
91     return -1;
92 }
93
94 public Object get(int index)
95 /*: requires "init & 0 <= index & index < csize"
96    ensures "(index, result) : contents & (ALL i v1 v2. (i, v1) : contents & (i,
97     v2) : contents --> v1 = v2)" */
98 {
99     return elementData[index];
100 }

```

```

101 public Object set(int index, Object element)
102 /*: requires "init & 0 <= index & index < csize"
103    modifies "contents"
104    ensures "(index, result) : old contents & contents = (old contents - {(index,
105        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
116        contents))) & ((index, element) : contents) & (ALL j e. (index < j & j <
117        csize --> ((j, e) : contents) = ((j - 1, e) : old contents))) & (csize =
118        (old csize) + 1) & (msize >= (old msize)) & (csize <= msize) & (ALL i v. (i,
119        v) : contents --> 0 <= i & i < csize) & (ALL i v1 v2. (i, v1) : contents &
120        (i, v2) : contents --> v1 = v2)" */
121 {
122     _ensureCapacity(size + 1);
123     int i = size - 1;
124     while /*: invariant "index - 1 <= i & i <= size - 1 & (ALL a j. ((a ~: hidden)
125         --> a.[j] = arrayRead (old Array.arrayState) a j)) & ((i = size - 1) -->
126         (ALL j. ((0 <= j & j < size) --> elementData.[j] ~: hidden))) & ((i < size -
127         1) --> (ALL j. ((0 <= j & j < size + 1) --> elementData.[j] ~: hidden))) &
128         ((ALL j. ((0 <= j & j <= i) --> elementData.[j] = arrayRead (old
129         Array.arrayState) (old elementData) j))) & ((ALL j. ((0 <= j & j <= index &
130         j < size) --> elementData.[j] = arrayRead (old Array.arrayState) (old
131         elementData) j))) & ((ALL j. ((i < j & j < size) --> elementData.[j + 1] =
132         arrayRead (old Array.arrayState) (old elementData) j))) & (ALL x i. (x ~:
133         elementData & x ~: (old elementData) --> (x.[i] = (arrayRead (old
134         Array.arrayState) (old x) i)))" */ (index <= i) {
135         elementData[i + 1] = elementData[i];
136         i = i - 1;
137     }
138     elementData[index] = element;
139     size = size + 1;
140     /*: note InvMeans: "(ALL j. ((0 <= j & j < index & j < (old size)) -->
141         elementData.[j] = arrayRead (old Array.arrayState) (old elementData) j)) &
142         (ALL j. ((index <= j & j < (old size)) --> elementData.[j + 1] = arrayRead
143         (old Array.arrayState) (old elementData) j))"; */
144     /*: note IndexRange: "0 <= index & index <= old size"; */
145     /*: note CsizeIsSize: "csize = size"; */
146     /*: note PostCond: "(ALL j e. (0 <= j & j < index --> ((j, e) : contents) =
147         ((j, e) : old contents)) & (index < j & j < csize --> ((j, e) : contents) =
148         ((j - 1, e) : old contents))" from InvMeans, PostCond, contents_def,
149         IndexRange, CsizeIsSize; */
150     /*: note OtherPostCond: "(index, element) : contents" from OtherPostCond,
151         contents_def, IndexRange; */
152 }
153
154 public Object remove_at(int index)
155 /*: requires "init & 0 <= index & index < csize"
156    modifies "contents", "csize"
157    ensures "((index, result) : old contents) & (csize = old csize - 1) & (ALL j e.
158        ((0 <= j & j < index) --> ((j, e) : contents) = ((j, e) : old contents)) &
159        ((index <= j & j < csize) --> ((j, e) : contents) = ((j + 1, e) : old
160        contents))) & (ALL i v. (i, v) : contents --> 0 <= i & i < csize)" */
161 {
162     Object oldValue = elementData[index];
163     int i = index;

```

```

139 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) {
140     elementData[i] = elementData[i+1];
141     i = i + 1;
142 }
143 size = size - 1;
144 elementData[size] = null;
145
146 return oldValue;
147 }
148 }

```

## 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

```

1 class ArrayListComm {
2     static void add_at_add_at_pre_s_0(ArrayList sa, ArrayList sb, int i1, Object v1,
3         int i2, Object v2)
4     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5         sa..contents = sb..contents & sa..csize = sb..csize"
6         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
7         "sb..msize"
8         ensures "True" */
9     {
10        /*: assume "(i1 < i2 & i2 <= sa..csize & (i2 - 1, v2) : sa..contents & 0 <= i2
11            - 1 & i2 - 1 < sa..csize) | (i1 = i2 & v1 = v2) | (i1 > i2 & (i1 - 1, v1) :
12            sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize)" */
13        /*: assume "0 <= i1 & i1 <= sa..csize" */
14        sa.add_at(i1, v1);
15        /*: assume "0 <= i2 & i2 <= sa..csize" */
16        sa.add_at(i2, v2);
17
18        sb.add_at(i2, v2);
19        sb.add_at(i1, v1);
20
21        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
22    }
23
24    static void add_at_add_at_pre_c_0(ArrayList sa, ArrayList sb, int i1, Object v1,
25        int i2, Object v2)
26    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
27        sa..contents = sb..contents & sa..csize = sb..csize"
28        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
29        "sb..msize"
30        ensures "True" */
31    {
32        /*: assume "~((i1 < i2 & i2 <= sa..csize & (i2 - 1, v2) : sa..contents & 0 <=
33            i2 - 1 & i2 - 1 < sa..csize) | (i1 = i2 & v1 = v2) | (i1 > i2 & (i1 - 1, v1)
34            : sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize))" */
35        /*: assume "0 <= i1 & i1 <= sa..csize" */
36        sa.add_at(i1, v1);
37        /*: assume "0 <= i2 & i2 <= sa..csize" */
38        sa.add_at(i2, v2);
39
40        /*: assume "0 <= i2 & i2 <= sb..csize" */
41        sb.add_at(i2, v2);

```

```

34     /*: assume "0 <= i1 & i1 <= sb..csize" */
35     sb.add_at(i1, v1);
36
37     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
38 }
39
40 static void add_at_add_at_between_s_1(ArrayList sa, ArrayList sb, int i1, Object
41     v1, int i2, Object v2)
42 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
43     sa..contents = sb..contents & sa..csize = sb..csize"
44     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
45     "sb..msize"
46     ensures "True" */
47 {
48     /*: assume "0 <= i1 & i1 <= sa..csize" */
49     sa.add_at(i1, v1);
50     /*: assume "(i1 < i2 & i2 <= sa..csize - 1 & (i2, v2) : sa..contents & 0 <= i2
51     & i2 < sa..csize) | (i1 = i2 & v1 = v2) | (i1 > i2 & (i1 - 1, v1) :
52     sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize)" */
53     /*: assume "0 <= i2 & i2 <= sa..csize" */
54     sa.add_at(i2, v2);
55
56     sb.add_at(i2, v2);
57     sb.add_at(i1, v1);
58
59     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
60 }
61
62 static void add_at_add_at_between_c_1(ArrayList sa, ArrayList sb, int i1, Object
63     v1, int i2, Object v2)
64 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
65     sa..contents = sb..contents & sa..csize = sb..csize"
66     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
67     "sb..msize"
68     ensures "True" */
69 {
70     /*: assume "0 <= i1 & i1 <= sa..csize" */
71     sa.add_at(i1, v1);
72     /*: assume "~((i1 < i2 & i2 <= sa..csize - 1 & (i2, v2) : sa..contents & 0 <=
73     i2 & i2 < sa..csize) | (i1 = i2 & v1 = v2) | (i1 > i2 & (i1 - 1, v1) :
74     sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize))" */
75     /*: assume "0 <= i2 & i2 <= sa..csize" */
76     sa.add_at(i2, v2);
77
78     /*: assume "0 <= i2 & i2 <= sb..csize" */
79     sb.add_at(i2, v2);
80     /*: assume "0 <= i1 & i1 <= sb..csize" */
81     sb.add_at(i1, v1);
82
83     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
84 }
85
86 static void add_at_add_at_post_s_2(ArrayList sa, ArrayList sb, int i1, Object v1,
87     int i2, Object v2)
88 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
89     sa..contents = sb..contents & sa..csize = sb..csize"
90     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
91     "sb..msize"
92     ensures "True" */
93 {
94     /*: assume "0 <= i1 & i1 <= sa..csize" */
95     sa.add_at(i1, v1);
96     /*: assume "0 <= i2 & i2 <= sa..csize" */
97     sa.add_at(i2, v2);

```

```

88     /*: assume "(i1 < i2 & i2 <= sa..csize - 2 & (i2 + 1, v2) : sa..contents & 0 <=
89         i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & v1 = v2) | (i1 > i2 & (i1, v1) :
90         sa..contents & 0 <= i1 & i1 < sa..csize)" */
91     sb.add_at(i2, v2);
92     sb.add_at(i1, v1);
93
94     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
95 }
96
97 static void add_at_add_at_post_c_2(ArrayList sa, ArrayList sb, int i1, Object v1,
98     int i2, Object v2)
99 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
100     sa..contents = sb..contents & sa..csize = sb..csize"
101     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
102     "sb..msize"
103     ensures "True" */
104 {
105     /*: assume "0 <= i1 & i1 <= sa..csize" */
106     sa.add_at(i1, v1);
107     /*: assume "0 <= i2 & i2 <= sa..csize" */
108     sa.add_at(i2, v2);
109     /*: assume "~((i1 < i2 & i2 <= sa..csize - 2 & (i2 + 1, v2) : sa..contents & 0
110         <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & v1 = v2) | (i1 > i2 & (i1, v1)
111         : sa..contents & 0 <= i1 & i1 < sa..csize))" */
112
113     /*: assume "0 <= i2 & i2 <= sb..csize" */
114     sb.add_at(i2, v2);
115     /*: assume "0 <= i1 & i1 <= sb..csize" */
116     sb.add_at(i1, v1);
117
118     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
119 }
120
121 static void add_at_get_pre_s_3(ArrayList sa, ArrayList sb, int i1, Object v1, int
122     i2)
123 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
124     sa..contents = sb..contents & sa..csize = sb..csize"
125     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
126     "sb..msize"
127     ensures "True" */
128 {
129     /*: assume "(i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents) =
130         ((i2, v) : sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2
131         < sa..csize) | (i1 = i2 & i2 < sa..csize & (i1, v1) : sa..contents & 0 <= i1
132         & i1 < sa..csize) | i1 > i2" */
133     /*: assume "0 <= i1 & i1 <= sa..csize" */
134     sa.add_at(i1, v1);
135     /*: assume "0 <= i2 & i2 < sa..csize" */
136     Object r2a = sa.get(i2);
137
138     Object r2b = sb.get(i2);
139     sb.add_at(i1, v1);
140
141     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
142 }
143
144 static void add_at_get_pre_c_3(ArrayList sa, ArrayList sb, int i1, Object v1, int
145     i2)
146 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
147     sa..contents = sb..contents & sa..csize = sb..csize"
148     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
149     "sb..msize"
150     ensures "True" */
151 {

```

```

140     /*: 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)" */
141     /*: assume "0 <= i1 & i1 <= sa..csize" */
142     sa.add_at(i1, v1);
143     /*: assume "0 <= i2 & i2 < sa..csize" */
144     Object r2a = sa.get(i2);
145
146     /*: assume "0 <= i2 & i2 < sb..csize" */
147     Object r2b = sb.get(i2);
148     /*: assume "0 <= i1 & i1 <= sb..csize" */
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_get_between_s_4(ArrayList sa, ArrayList sb, int i1, Object v1,
      int i2)
155 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
      sa..contents = sb..contents & sa..csize = sb..csize"
156 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
      "sb..msize"
157 ensures "True" */
158 {
159     /*: assume "0 <= i1 & i1 <= sa..csize" */
160     sa.add_at(i1, v1);
161     /*: 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" */
162     /*: assume "0 <= i2 & i2 < sa..csize" */
163     Object r2a = sa.get(i2);
164
165     Object r2b = sb.get(i2);
166     sb.add_at(i1, v1);
167
168     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
169 }
170
171
172 static void add_at_get_between_c_4(ArrayList sa, ArrayList sb, int i1, Object v1,
      int i2)
173 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
      sa..contents = sb..contents & sa..csize = sb..csize"
174 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
      "sb..msize"
175 ensures "True" */
176 {
177     /*: assume "0 <= i1 & i1 <= sa..csize" */
178     sa.add_at(i1, v1);
179     /*: 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)" */
180     /*: assume "0 <= i2 & i2 < sa..csize" */
181     Object r2a = sa.get(i2);
182
183     Object r2b = sb.get(i2);
184     /*: assume "0 <= i2 & i2 < sb..csize" */
185     Object r2b = sb.get(i2);
186     /*: assume "0 <= i1 & i1 <= sb..csize" */
187     sb.add_at(i1, v1);
188
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
193     i2)
194 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
195     sa..contents = sb..contents & sa..csize = sb..csize"
196     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
197     "sb..msize"
198     ensures "True" */
199 {
200     /*: assume "0 <= i1 & i1 <= sa..csize" */
201     sa.add_at(i1, v1);
202     /*: assume "0 <= i2 & i2 < sa..csize" */
203     Object r2a = sa.get(i2);
204     /*: assume "(i1 < i2 & i2 < sa..csize - 1 & (i2 + 1, r2a) : sa..contents & 0 <=
205         i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, v1)
206         : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | i1 > i2" */
207
208     Object r2b = sb.get(i2);
209     sb.add_at(i1, v1);
210
211     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
212 }
213
214 static void add_at_get_post_c_5(ArrayList sa, ArrayList sb, int i1, Object v1, int
215     i2)
216 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
217     sa..contents = sb..contents & sa..csize = sb..csize"
218     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
219     "sb..msize"
220     ensures "True" */
221 {
222     /*: assume "0 <= i1 & i1 <= sa..csize" */
223     sa.add_at(i1, v1);
224     /*: assume "0 <= i2 & i2 < sa..csize" */
225     Object r2a = sa.get(i2);
226     /*: assume "~((i1 < i2 & i2 < sa..csize - 1 & (i2 + 1, r2a) : sa..contents & 0
227         <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1,
228         v1) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | i1 > i2)" */
229
230     /*: assume "0 <= i2 & i2 < sb..csize" */
231     Object r2b = sb.get(i2);
232     /*: assume "0 <= i1 & i1 <= sb..csize" */
233     sb.add_at(i1, v1);
234
235     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
236         */
237 }
238
239 static void add_at_indexOf_pre_s_6(ArrayList sa, ArrayList sb, int i1, Object v1,
240     Object v2)
241 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
242     sa..contents = sb..contents & sa..csize = sb..csize"
243     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
244     "sb..msize"
245     ensures "True" */
246 {
247     /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
248         v2) | (EX i. (i, v2) : sa..contents & 0 <= i & i < i1) | (~(EX i. (i, v2) :
249         sa..contents & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 <
250         sa..csize & v1 = v2)" */
251     /*: assume "0 <= i1 & i1 <= sa..csize" */
252     sa.add_at(i1, v1);
253     int r2a = sa.indexOf(v2);
254

```



```

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

```

```

291         & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize &
292         (i1 + 1, v2) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
293     int r2a = sa.index0f(v2);
294
295     int r2b = sb.index0f(v2);
296     /*: assume "0 <= i1 & i1 <= sb..csize" */
297     sb.add_at(i1, v1);
298
299     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
300     */
301 }
302
303 static void add_at_index0f_post_s_8(ArrayList sa, ArrayList sb, int i1, Object v1,
304     Object v2)
305 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
306     sa..contents = sb..contents & sa..csize = sb..csize"
307 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
308     "sb..msize"
309 ensures "True" */
310 {
311     /*: assume "0 <= i1 & i1 <= sa..csize" */
312     sa.add_at(i1, v1);
313     int r2a = sa.index0f(v2);
314     /*: assume "r2a < 0 | (0 <= r2a & r2a < i1) | (r2a = i1 & (i1 + 1, v2) :
315     sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
316
317     int r2b = sb.index0f(v2);
318     sb.add_at(i1, v1);
319
320     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
321 }
322
323 static void add_at_index0f_post_c_8(ArrayList sa, ArrayList sb, int i1, Object v1,
324     Object v2)
325 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
326     sa..contents = sb..contents & sa..csize = sb..csize"
327 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
328     "sb..msize"
329 ensures "True" */
330 {
331     /*: assume "0 <= i1 & i1 <= sa..csize" */
332     sa.add_at(i1, v1);
333     int r2a = sa.index0f(v2);
334     /*: assume "~(r2a < 0 | (0 <= r2a & r2a < i1) | (r2a = i1 & (i1 + 1, v2) :
335     sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
336
337     int r2b = sb.index0f(v2);
338     /*: assume "0 <= i1 & i1 <= sb..csize" */
339     sb.add_at(i1, v1);
340
341     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
342     */
343 }
344
345 static void add_at_lastIndex0f_pre_s_9(ArrayList sa, ArrayList sb, int i1, Object
346     v1, Object v2)
347 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
348     sa..contents = sb..contents & sa..csize = sb..csize"
349 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
350     "sb..msize"
351 ensures "True" */
352 {
353     /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
354     v2) | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
355     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",
356     "sb..msize"
357     ensures "True" */
358 {
359     /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
360     v2) | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
361     sa..contents & i1 <= i & i < sa..csize) & v1 ~= v2))" */
362     /*: assume "0 <= i1 & i1 <= sa..csize" */
363     sa.add_at(i1, v1);
364     int r2a = sa.lastIndexOf(v2);
365
366     int r2b = sb.lastIndexOf(v2);
367     /*: assume "0 <= i1 & i1 <= sb..csize" */
368     sb.add_at(i1, v1);
369
370     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
371     */
372 }
373
374 static void add_at_lastIndexOf_between_s_10(ArrayList sa, ArrayList sb, int i1,
    Object v1, Object v2)
375 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
376     sa..contents = sb..contents & sa..csize = sb..csize"
377     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
378     "sb..msize"
379     ensures "True" */
380 {
381     /*: assume "0 <= i1 & i1 <= sa..csize" */
382     sa.add_at(i1, v1);
383     /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | ((EX i.
384     (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) : sa..contents &
385     i1 <= i & i < sa..csize))" */
386     int r2a = sa.lastIndexOf(v2);
387
388     int r2b = sb.lastIndexOf(v2);
389     sb.add_at(i1, v1);
390
391     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
392 }
393
394 static void add_at_lastIndexOf_between_c_10(ArrayList sa, ArrayList sb, int i1,
    Object v1, Object v2)
395 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
396     sa..contents = sb..contents & sa..csize = sb..csize"
397     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
398     "sb..msize"
399     ensures "True" */
400 {
401     /*: assume "0 <= i1 & i1 <= sa..csize" */
402     sa.add_at(i1, v1);

```

```

395     /*: assume "~((EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | ((EX
396       i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
397         sa..contents & i1 <= i & i < sa..csize)))" */
398     int r2a = sa.lastIndexOf(v2);
399
400     int r2b = sb.lastIndexOf(v2);
401     /*: assume "0 <= i1 & i1 <= sb..csize" */
402     sb.add_at(i1, v1);
403
404     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
405       */
406 }
407
408 static void add_at_lastIndexOf_post_s_11(ArrayList sa, ArrayList sb, int i1, Object
409   v1, Object v2)
410 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
411   sa..contents = sb..contents & sa..csize = sb..csize"
412   modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
413   "sb..msize"
414   ensures "True" */
415 {
416   /*: assume "0 <= i1 & i1 <= sa..csize" */
417   sa.add_at(i1, v1);
418   int r2a = sa.lastIndexOf(v2);
419   /*: assume "r2a < 0 | (0 <= r2a & r2a < i1)" */
420
421   int r2b = sb.lastIndexOf(v2);
422   sb.add_at(i1, v1);
423
424   /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
425 }
426
427 static void add_at_lastIndexOf_post_c_11(ArrayList sa, ArrayList sb, int i1, Object
428   v1, Object v2)
429 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
430   sa..contents = sb..contents & sa..csize = sb..csize"
431   modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
432   "sb..msize"
433   ensures "True" */
434 {
435   /*: assume "0 <= i1 & i1 <= sa..csize" */
436   sa.add_at(i1, v1);
437   int r2a = sa.lastIndexOf(v2);
438   /*: assume "~(r2a < 0 | (0 <= r2a & r2a < i1))" */
439
440   int r2b = sb.lastIndexOf(v2);
441   /*: assume "0 <= i1 & i1 <= sb..csize" */
442   sb.add_at(i1, v1);
443
444   /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
445     */
446 }
447
448 static void add_at_remove_at_pre_s_12(ArrayList sa, ArrayList sb, int i1, Object
449   v1, int i2)
450 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
451   sa..contents = sb..contents & sa..csize = sb..csize"
452   modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
453   "sb..msize"
454   ensures "True" */
455 {
456   /*: assume "(i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents) =
457     ((i2, v) : sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2
458     < sa..csize) | (sa..csize - 1 >= i1 & i1 = i2 & (i1, v1) : sa..contents & 0

```

```

447     <= i1 & i1 < sa..csize) | (sa..csize - 1 >= i1 & i1 > i2 & (i1, v1) :
448     sa..contents & 0 <= i1 & i1 < sa..csize)" */
449     /*: assume "0 <= i1 & i1 <= sa..csize" */
450     sa.add_at(i1, v1);
451     /*: assume "0 <= i2 & i2 < sa..csize" */
452     Object r2a = sa.remove_at(i2);
453     Object r2b = sb.remove_at(i2);
454     sb.add_at(i1, v1);
455     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
456 }
457
458 static void add_at_remove_at_pre_c_12(ArrayList sa, ArrayList sb, int i1, Object
459 v1, int i2)
460 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
461     sa..contents = sb..contents & sa..csize = sb..csize"
462     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
463     "sb..msize"
464     ensures "True" */
465 {
466     /*: assume "~((i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents)
467     = ((i2, v) : sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 &
468     i2 < sa..csize) | (sa..csize - 1 >= i1 & i1 = i2 & (i1, v1) : sa..contents &
469     0 <= i1 & i1 < sa..csize) | (sa..csize - 1 >= i1 & i1 > i2 & (i1, v1) :
470     sa..contents & 0 <= i1 & i1 < sa..csize))" */
471     /*: assume "0 <= i1 & i1 <= sa..csize" */
472     sa.add_at(i1, v1);
473     /*: assume "0 <= i2 & i2 < sa..csize" */
474     Object r2a = sa.remove_at(i2);
475     /*: assume "0 <= i2 & i2 < sb..csize" */
476     Object r2b = sb.remove_at(i2);
477     /*: assume "0 <= i1 & i1 <= sb..csize" */
478     sb.add_at(i1, v1);
479     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
480     */
481 }
482
483 static void add_at_remove_at_between_s_13(ArrayList sa, ArrayList sb, int i1,
484 Object v1, int i2)
485 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
486     sa..contents = sb..contents & sa..csize = sb..csize"
487     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
488     "sb..msize"
489     ensures "True" */
490 {
491     /*: assume "0 <= i1 & i1 <= sa..csize" */
492     sa.add_at(i1, v1);
493     /*: assume "(i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa..contents) =
494     ((i2 + 1, v) : sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2
495     + 1 < sa..csize) | (sa..csize - 2 >= i1 & i1 = i2 & (i1 + 1, v1) :
496     sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 2 >= i1 & i1
497     > i2 & (i1 + 1, v1) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
498     /*: assume "0 <= i2 & i2 < sa..csize" */
499     Object r2a = sa.remove_at(i2);
500     Object r2b = sb.remove_at(i2);
501     sb.add_at(i1, v1);
502     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
503 }

```

```

496 static void add_at_remove_at_between_c_13(ArrayList sa, ArrayList sb, int i1,
497     Object v1, int i2)
498 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
499     sa..contents = sb..contents & sa..csize = sb..csize"
500     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
501     "sb..msize"
502     ensures "True" */
503 {
504     /*: assume "0 <= i1 & i1 <= sa..csize" */
505     sa.add_at(i1, v1);
506     /*: assume "~((i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa..contents)
507     = ((i2 + 1, v) : sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 &
508     i2 + 1 < sa..csize) | (sa..csize - 2 >= i1 & i1 = i2 & (i1 + 1, v1) :
509     sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 2 >= i1 & i1
510     > i2 & (i1 + 1, v1) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
511     /*: assume "0 <= i2 & i2 < sa..csize" */
512     Object r2a = sa.remove_at(i2);
513
514     /*: assume "0 <= i2 & i2 < sb..csize" */
515     Object r2b = sb.remove_at(i2);
516     /*: assume "0 <= i1 & i1 <= sb..csize" */
517     sb.add_at(i1, v1);
518
519     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
520     */
521 }
522
523 static void add_at_remove_at_post_s_14(ArrayList sa, ArrayList sb, int i1, Object
524     v1, int i2)
525 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
526     sa..contents = sb..contents & sa..csize = sb..csize"
527     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
528     "sb..msize"
529     ensures "True" */
530 {
531     /*: assume "0 <= i1 & i1 <= sa..csize" */
532     sa.add_at(i1, v1);
533     /*: assume "0 <= i2 & i2 < sa..csize" */
534     Object r2a = sa.remove_at(i2);
535     /*: assume "(i1 < i2 & i2 < sa..csize & (i2, r2a) : sa..contents & 0 <= i2 & i2
536     < sa..csize) | (sa..csize - 1 >= i1 & i1 = i2 & (i1, v1) : sa..contents & 0
537     <= i1 & i1 < sa..csize) | (sa..csize - 1 >= i1 & i1 > i2 & (i1, v1) :
538     sa..contents & 0 <= i1 & i1 < sa..csize)" */
539
540     Object r2b = sb.remove_at(i2);
541     sb.add_at(i1, v1);
542
543     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
544 }
545
546 static void add_at_remove_at_post_c_14(ArrayList sa, ArrayList sb, int i1, Object
547     v1, int i2)
548 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
549     sa..contents = sb..contents & sa..csize = sb..csize"
550     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
551     "sb..msize"
552     ensures "True" */
553 {
554     /*: assume "0 <= i1 & i1 <= sa..csize" */
555     sa.add_at(i1, v1);
556     /*: assume "0 <= i2 & i2 < sa..csize" */
557     Object r2a = sa.remove_at(i2);
558     /*: assume "~((i1 < i2 & i2 < sa..csize & (i2, r2a) : sa..contents & 0 <= i2 &
559     i2 < sa..csize) | (sa..csize - 1 >= i1 & i1 = i2 & (i1, v1) : sa..contents &

```

```

545         0 <= i1 & i1 < sa..csize) | (sa..csize - 1 >= i1 & i1 > i2 & (i1, v1) :
546         sa..contents & 0 <= i1 & i1 < sa..csize))" */
547
548     /*: assume "0 <= i2 & i2 < sb..csize" */
549     Object r2b = sb.remove_at(i2);
550     /*: assume "0 <= i1 & i1 <= sb..csize" */
551     sb.add_at(i1, v1);
552
553     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
554     */
555 }
556
557 static void add_at_remove_at_pre_s_15(ArrayList sa, ArrayList sb, int i1, Object
558     v1, int i2)
559 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
560     sa..contents = sb..contents & sa..csize = sb..csize"
561     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
562     "sb..msize"
563     ensures "True" */
564 {
565     /*: assume "(i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents) =
566     ((i2, v) : sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2
567     < sa..csize) | (sa..csize - 1 >= i1 & i1 = i2 & (i1, v1) : sa..contents & 0
568     <= i1 & i1 < sa..csize) | (sa..csize - 1 >= i1 & i1 > i2 & (i1, v1) :
569     sa..contents & 0 <= i1 & i1 < sa..csize)" */
570     /*: assume "0 <= i1 & i1 <= sa..csize" */
571     sa.add_at(i1, v1);
572     /*: assume "0 <= i2 & i2 < sa..csize" */
573     sa.remove_at(i2);
574
575     sb.remove_at(i2);
576     sb.add_at(i1, v1);
577
578     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
579 }
580
581 static void add_at_remove_at_pre_c_15(ArrayList sa, ArrayList sb, int i1, Object
582     v1, int i2)
583 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
584     sa..contents = sb..contents & sa..csize = sb..csize"
585     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
586     "sb..msize"
587     ensures "True" */
588 {
589     /*: assume "~((i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents)
590     = ((i2, v) : sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 &
591     i2 < sa..csize) | (sa..csize - 1 >= i1 & i1 = i2 & (i1, v1) : sa..contents &
592     0 <= i1 & i1 < sa..csize) | (sa..csize - 1 >= i1 & i1 > i2 & (i1, v1) :
593     sa..contents & 0 <= i1 & i1 < sa..csize))" */
594     /*: assume "0 <= i1 & i1 <= sa..csize" */
595     sa.add_at(i1, v1);
596     /*: assume "0 <= i2 & i2 < sa..csize" */
597     sa.remove_at(i2);
598
599     /*: assume "0 <= i2 & i2 < sb..csize" */
600     sb.remove_at(i2);
601     /*: assume "0 <= i1 & i1 <= sb..csize" */
602     sb.add_at(i1, v1);
603
604     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
605 }
606
607 static void add_at_remove_at_between_s_16(ArrayList sa, ArrayList sb, int i1,
608     Object v1, int i2)
609 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &

```

```

594         sa..contents = sb..contents & sa..csize = sb..csize"
595 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
        "sb..msize"
596 ensures "True" */
597 {
598     /*: assume "0 <= i1 & i1 <= sa..csize" */
599     sa.add_at(i1, v1);
600     /*: 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)" */
601     /*: assume "0 <= i2 & i2 < sa..csize" */
602     sa.remove_at(i2);
603
604     sb.remove_at(i2);
605     sb.add_at(i1, v1);
606
607     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
608 }
609
610 static void add_at_remove_at_between_c_16(ArrayList sa, ArrayList sb, int i1,
        Object v1, int i2)
611 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
612 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
        "sb..msize"
613 ensures "True" */
614 {
615     /*: assume "0 <= i1 & i1 <= sa..csize" */
616     sa.add_at(i1, v1);
617     /*: 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))" */
618     /*: assume "0 <= i2 & i2 < sa..csize" */
619     sa.remove_at(i2);
620
621     /*: assume "0 <= i2 & i2 < sb..csize" */
622     sb.remove_at(i2);
623     /*: assume "0 <= i1 & i1 <= sb..csize" */
624     sb.add_at(i1, v1);
625
626     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
627 }
628
629 static void add_at_remove_at_post_s_17(ArrayList sa, ArrayList sb, int i1, Object
        v1, int i2)
630 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
631 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
        "sb..msize"
632 ensures "True" */
633 {
634     /*: assume "0 <= i1 & i1 <= sa..csize" */
635     sa.add_at(i1, v1);
636     /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
637     /*: ghost specvar sa__csize :: "int" = "sa..csize" */
638     /*: assume "0 <= i2 & i2 < sa..csize" */
639     sa.remove_at(i2);
640     /*: 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)" */
643
        sb.remove_at(i2);
644
        sb.add_at(i1, v1);
645
646
        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
647
    }
648
649
static void add_at_remove_at_post_c_17(ArrayList sa, ArrayList sb, int i1, Object
        v1, int i2)
650
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
651     sa..contents = sb..contents & sa..csize = sb..csize"
652     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
653     "sb..msize"
654     ensures "True" */
655
{
656
    /*: assume "0 <= i1 & i1 <= sa..csize" */
657
    sa.add_at(i1, v1);
658
    /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
659
    /*: ghost specvar sa__csize :: "int" = "sa..csize" */
660
    /*: assume "0 <= i2 & i2 < sa..csize" */
661
    sa.remove_at(i2);
662
    /*: 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))" */
663
664
    /*: assume "0 <= i2 & i2 < sb..csize" */
665
    sb.remove_at(i2);
666
    /*: assume "0 <= i1 & i1 <= sb..csize" */
667
    sb.add_at(i1, v1);
668
669
    /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
670
}
671
672
static void add_at_set_pre_s_18(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
673
/*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
674     sa..contents = sb..contents & sa..csize = sb..csize"
675     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
676     "sb..msize"
677     ensures "True" */
678
{
679
    /*: 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" */
680
    /*: assume "0 <= i1 & i1 <= sa..csize" */
681
    sa.add_at(i1, v1);
682
    /*: assume "0 <= i2 & i2 < sa..csize" */
683
    Object r2a = sa.set(i2, v2);
684
685
    Object r2b = sb.set(i2, v2);
686
    sb.add_at(i1, v1);
687
688
    /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
689
}
690
691
static void add_at_set_pre_c_18(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
692
/*: 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
749 static void add_at_set_post_s_20(ArrayList sa, ArrayList sb, int i1, Object v1, int
750     i2, Object v2)
751 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
752     sa..contents = sb..contents & sa..csize = sb..csize"
753     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
754     "sb..msize"
755     ensures "True" */
756 {
757     /*: assume "0 <= i1 & i1 <= sa..csize" */
758     sa.add_at(i1, v1);
759     /*: assume "0 <= i2 & i2 < sa..csize" */
760     Object r2a = sa.set(i2, v2);
761     /*: assume "(i1 < i2 & i2 < sa..csize - 1 & (i2 + 1, r2a) : sa..contents & r2a
762     = v2 & (i2 + 1, v2) : sa..contents & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1
763     = i2 & i2 < sa..csize - 1 & (i2 + 1, v1) : sa..contents & (i2 + 1, v2) :
764     sa..contents & v1 = v2 & 0 <= i2 + 1 & i2 + 1 < sa..csize) | i1 > i2" */
765
766     Object r2b = sb.set(i2, v2);
767     sb.add_at(i1, v1);
768
769     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
770 }
771
772 static void add_at_set_post_c_20(ArrayList sa, ArrayList sb, int i1, Object v1, int
773     i2, Object v2)
774 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
775     sa..contents = sb..contents & sa..csize = sb..csize"
776     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
777     "sb..msize"
778     ensures "True" */
779 {
780     /*: assume "0 <= i1 & i1 <= sa..csize" */
781     sa.add_at(i1, v1);
782     /*: assume "0 <= i2 & i2 < sa..csize" */
783     Object r2a = sa.set(i2, v2);
784     /*: assume "~((i1 < i2 & i2 < sa..csize - 1 & (i2 + 1, r2a) : sa..contents &
785     r2a = v2 & (i2 + 1, v2) : sa..contents & 0 <= i2 + 1 & i2 + 1 < sa..csize) |
786     (i1 = i2 & i2 < sa..csize - 1 & (i2 + 1, v1) : sa..contents & (i2 + 1, v2) :
787     sa..contents & v1 = v2 & 0 <= i2 + 1 & i2 + 1 < sa..csize) | i1 > i2)" */
788
789     /*: assume "0 <= i2 & i2 < sb..csize" */
790     Object r2b = sb.set(i2, v2);
791     /*: assume "0 <= i1 & i1 <= sb..csize" */
792     sb.add_at(i1, v1);
793
794     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
795     */
796 }
797
798 static void add_at_set_pre_s_21(ArrayList sa, ArrayList sb, int i1, Object v1, int
799     i2, Object v2)
800 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
801     sa..contents = sb..contents & sa..csize = sb..csize"
802     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
803     "sb..msize"
804     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 &
      sa..contents = sb..contents & sa..csize = sb..csize"
806     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
      "sb..msize"
807     ensures "True" */
808 {
809     /*: 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)" */
810     /*: assume "0 <= i1 & i1 <= sa..csize" */
811     sa.add_at(i1, v1);
812     /*: assume "0 <= i2 & i2 < sa..csize" */
813     sa.set(i2, v2);
814
815     /*: assume "0 <= i2 & i2 < sb..csize" */
816     sb.set(i2, v2);
817     /*: assume "0 <= i1 & i1 <= sb..csize" */
818     sb.add_at(i1, v1);
819
820     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
821 }
822
823 static void add_at_set_between_s_22(ArrayList sa, ArrayList sb, int i1, Object v1,
      int i2, Object v2)
824 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
      sa..contents = sb..contents & sa..csize = sb..csize"
825     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
      "sb..msize"
826     ensures "True" */
827 {
828     /*: assume "0 <= i1 & i1 <= sa..csize" */
829     sa.add_at(i1, v1);
830     /*: 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" */
831     /*: assume "0 <= i2 & i2 < sa..csize" */
832     sa.set(i2, v2);
833
834     sb.set(i2, v2);
835     sb.add_at(i1, v1);
836
837     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
838 }
839

```

```

840 }
841
842 static void add_at_set_between_c_22(ArrayList sa, ArrayList sb, int i1, Object v1,
843     int i2, Object v2)
844 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
845     sa..contents = sb..contents & sa..csize = sb..csize"
846     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
847     "sb..msize"
848     ensures "True" */
849 {
850     /*: assume "0 <= i1 & i1 <= sa..csize" */
851     sa.add_at(i1, v1);
852     /*: assume "~((i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa..contents)
853     = ((i2 + 1, v) : sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) :
854     sa..contents & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize)
855     | (i1 = i2 & i2 < sa..csize - 1 & (i2 + 1, v1) : sa..contents & (i2 + 1, v2)
856     : sa..contents & v1 = v2 & 0 <= i2 + 1 & i2 + 1 < sa..csize) | i1 > i2)" */
857     /*: assume "0 <= i2 & i2 < sa..csize" */
858     sa.set(i2, v2);
859
860     /*: assume "0 <= i2 & i2 < sb..csize" */
861     sb.set(i2, v2);
862     /*: assume "0 <= i1 & i1 <= sb..csize" */
863     sb.add_at(i1, v1);
864
865     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
866 }
867
868 static void add_at_set_post_s_23(ArrayList sa, ArrayList sb, int i1, Object v1, int
869     i2, Object v2)
870 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
871     sa..contents = sb..contents & sa..csize = sb..csize"
872     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
873     "sb..msize"
874     ensures "True" */
875 {
876     /*: assume "0 <= i1 & i1 <= sa..csize" */
877     sa.add_at(i1, v1);
878     /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
879     /*: ghost specvar sa__csize :: "int" = "sa..csize" */
880     /*: assume "0 <= i2 & i2 < sa..csize" */
881     sa.set(i2, v2);
882     /*: assume "(i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa__contents) =
883     ((i2 + 1, v) : sa..contents)) & (i2, v2) : sa__contents & (i2 + 1, v2) :
884     sa..contents & 0 <= i2 & i2 < sa__csize & 0 <= i2 + 1 & i2 + 1 < sa..csize)
885     | (i1 = i2 & i2 < sa..csize - 1 & (i2 + 1, v1) : sa..contents & (i2 + 1, v2)
886     : sa..contents & v1 = v2 & 0 <= i2 + 1 & i2 + 1 < sa..csize) | i1 > i2" */
887
888     sb.set(i2, v2);
889     sb.add_at(i1, v1);
890
891     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
892 }
893
894 static void add_at_set_post_c_23(ArrayList sa, ArrayList sb, int i1, Object v1, int
895     i2, Object v2)
896 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
897     sa..contents = sb..contents & sa..csize = sb..csize"
898     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
899     "sb..msize"
900     ensures "True" */
901 {
902     /*: assume "0 <= i1 & i1 <= sa..csize" */
903     sa.add_at(i1, v1);
904     /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */

```

```

891     /*: ghost specvar sa_..csize :: "int" = "sa..csize" */
892     /*: assume "0 <= i2 & i2 < sa..csize" */
893     sa.set(i2, v2);
894     /*: 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)" */
895
896     /*: assume "0 <= i2 & i2 < sb..csize" */
897     sb.set(i2, v2);
898     /*: assume "0 <= i1 & i1 <= sb..csize" */
899     sb.add_at(i1, v1);
900
901     /*: assert "~(sa_..contents = sb_..contents & sa_..csize = sb_..csize)" */
902 }
903
904 static void add_at_size_pre_s_24(ArrayList sa, ArrayList sb, int i1, Object v1)
905 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa_..init & sb_..init &
906             sa_..contents = sb_..contents & sa_..csize = sb_..csize"
907 modifies "sa_..contents", "sb_..contents", "sa_..csize", "sb_..csize", "sa_..msize",
908           "sb_..msize"
909 ensures "True" */
910 {
911     /*: assume "False" */
912     /*: assume "0 <= i1 & i1 <= sa_..csize" */
913     sa.add_at(i1, v1);
914     int r2a = sa.size();
915
916     int r2b = sb.size();
917     sb.add_at(i1, v1);
918
919     /*: assert "r2a = r2b & sa_..contents = sb_..contents & sa_..csize = sb_..csize" */
920 }
921
922 static void add_at_size_pre_c_24(ArrayList sa, ArrayList sb, int i1, Object v1)
923 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa_..init & sb_..init &
924             sa_..contents = sb_..contents & sa_..csize = sb_..csize"
925 modifies "sa_..contents", "sb_..contents", "sa_..csize", "sb_..csize", "sa_..msize",
926           "sb_..msize"
927 ensures "True" */
928 {
929     /*: assume "~(False)" */
930     /*: assume "0 <= i1 & i1 <= sa_..csize" */
931     sa.add_at(i1, v1);
932     int r2a = sa.size();
933
934     int r2b = sb.size();
935     /*: assume "0 <= i1 & i1 <= sb_..csize" */
936     sb.add_at(i1, v1);
937
938     /*: assert "~(r2a = r2b & sa_..contents = sb_..contents & sa_..csize = sb_..csize)"
939           */
940 }
941
942 static void add_at_size_between_s_25(ArrayList sa, ArrayList sb, int i1, Object v1)
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", "sa_..msize",
946           "sb_..msize"
947 ensures "True" */
948 {
949     /*: assume "0 <= i1 & i1 <= sa_..csize" */
950     sa.add_at(i1, v1);
951     /*: 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",
960         "sb..msize"
961 ensures "True" */
962 {
963     /*: assume "0 <= i1 & i1 <= sa..csize" */
964     sa.add_at(i1, v1);
965     /*: assume "~(False)" */
966     int r2a = sa.size();
967
968     int r2b = sb.size();
969     /*: assume "0 <= i1 & i1 <= sb..csize" */
970     sb.add_at(i1, v1);
971
972     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
973             */
974 }
975
976 static void add_at_size_post_s_26(ArrayList sa, ArrayList sb, int i1, Object v1)
977 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
978             sa..contents = sb..contents & sa..csize = sb..csize"
979 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
980         "sb..msize"
981 ensures "True" */
982 {
983     /*: assume "0 <= i1 & i1 <= sa..csize" */
984     sa.add_at(i1, v1);
985     int r2a = sa.size();
986     /*: assume "False" */
987
988     int r2b = sb.size();
989     sb.add_at(i1, v1);
990
991     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
992 }
993
994 static void add_at_size_post_c_26(ArrayList sa, ArrayList sb, int i1, Object v1)
995 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
996             sa..contents = sb..contents & sa..csize = sb..csize"
997 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
998         "sb..msize"
999 ensures "True" */
1000 {
1001     /*: assume "0 <= i1 & i1 <= sa..csize" */
1002     sa.add_at(i1, v1);
1003     int r2a = sa.size();
1004     /*: assume "~(False)" */
1005
1006     int r2b = sb.size();
1007     /*: assume "0 <= i1 & i1 <= sb..csize" */
1008     sb.add_at(i1, v1);
1009
1010     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
1011             */
1012 }

```

```

1008 static void get_add_at_pre_s_27(ArrayList sa, ArrayList sb, int i1, int i2, Object
1009     v2)
1010 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1011     sa..contents = sb..contents & sa..csize = sb..csize"
1012     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
1013     "sb..msize"
1014     ensures "True" */
1015 {
1016     /*: assume "i1 < i2 | (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 <
1017     sa..csize) | (i1 > i2 & (ALL v. ((i1 - 1, v) : sa..contents) = ((i1, v) :
1018     sa..contents)) & 0 <= i1 - 1 & i1 - 1 < sa..csize & 0 <= i1 & i1 <
1019     sa..csize)" */
1020     /*: assume "0 <= i1 & i1 < sa..csize" */
1021     Object r1a = sa.get(i1);
1022     /*: assume "0 <= i2 & i2 <= sa..csize" */
1023     sa.add_at(i2, v2);
1024     sb.add_at(i2, v2);
1025     Object r1b = sb.get(i1);
1026     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
1027 }
1028 static void get_add_at_pre_c_27(ArrayList sa, ArrayList sb, int i1, int i2, Object
1029     v2)
1030 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1031     sa..contents = sb..contents & sa..csize = sb..csize"
1032     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
1033     "sb..msize"
1034     ensures "True" */
1035 {
1036     /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 <
1037     sa..csize) | (i1 > i2 & (ALL v. ((i1 - 1, v) : sa..contents) = ((i1, v) :
1038     sa..contents)) & 0 <= i1 - 1 & i1 - 1 < sa..csize & 0 <= i1 & i1 <
1039     sa..csize))" */
1040     /*: assume "0 <= i1 & i1 < sa..csize" */
1041     Object r1a = sa.get(i1);
1042     /*: assume "0 <= i2 & i2 <= sa..csize" */
1043     sa.add_at(i2, v2);
1044     /*: assume "0 <= i2 & i2 <= sb..csize" */
1045     sb.add_at(i2, v2);
1046     /*: assume "0 <= i1 & i1 < sb..csize" */
1047     Object r1b = sb.get(i1);
1048     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
1049     */
1050 }
1051 static void get_add_at_between_s_28(ArrayList sa, ArrayList sb, int i1, int i2,
1052     Object v2)
1053 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1054     sa..contents = sb..contents & sa..csize = sb..csize"
1055     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
1056     "sb..msize"
1057     ensures "True" */
1058 {
1059     /*: assume "0 <= i1 & i1 < sa..csize" */
1060     Object r1a = sa.get(i1);
1061     /*: assume "i1 < i2 | (i1 = i2 & r1a = v2) | (i1 > i2 & (i1 - 1, r1a) :
1062     sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize)" */
1063     /*: assume "0 <= i2 & i2 <= sa..csize" */
1064     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,
1066     Object v2)
1067 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1068     sa..contents = sb..contents & sa..csize = sb..csize"
1069     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
1070     "sb..msize"
1071     ensures "True" */
1072 {
1073     /*: assume "0 <= i1 & i1 < sa..csize" */
1074     Object r1a = sa.get(i1);
1075     /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v2) | (i1 > i2 & (i1 - 1, r1a) :
1076     sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize))" */
1077     /*: assume "0 <= i2 & i2 <= sa..csize" */
1078     sa.add_at(i2, v2);
1079
1080     /*: assume "0 <= i2 & i2 <= sb..csize" */
1081     sb.add_at(i2, v2);
1082     /*: assume "0 <= i1 & i1 < sb..csize" */
1083     Object r1b = sb.get(i1);
1084
1085     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
1086     */
1087 }
1088
1089 static void get_add_at_post_s_29(ArrayList sa, ArrayList sb, int i1, int i2, Object
1090     v2)
1091 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1092     sa..contents = sb..contents & sa..csize = sb..csize"
1093     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
1094     "sb..msize"
1095     ensures "True" */
1096 {
1097     /*: assume "0 <= i1 & i1 < sa..csize" */
1098     Object r1a = sa.get(i1);
1099     /*: assume "0 <= i2 & i2 <= sa..csize" */
1100     sa.add_at(i2, v2);
1101     /*: assume "i1 < i2 | (i1 = i2 & r1a = v2) | (i1 > i2 & (i1, r1a) :
1102     sa..contents & 0 <= i1 & i1 < sa..csize)" */
1103
1104     sb.add_at(i2, v2);
1105     Object r1b = sb.get(i1);
1106
1107     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
1108 }
1109
1110 static void get_add_at_post_c_29(ArrayList sa, ArrayList sb, int i1, int i2, Object
1111     v2)
1112 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1113     sa..contents = sb..contents & sa..csize = sb..csize"
1114     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
1115     "sb..msize"
1116     ensures "True" */
1117 {
1118     /*: assume "0 <= i1 & i1 < sa..csize" */
1119     Object r1a = sa.get(i1);
1120     /*: assume "0 <= i2 & i2 <= sa..csize" */
1121     sa.add_at(i2, v2);
1122     /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v2) | (i1 > i2 & (i1, r1a) :
1123     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 =
1193          sb..csize)" */
1194 }
1195
1196 static void get_get_post_s_32(ArrayList sa, ArrayList sb, int i1, int i2)
1197 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1198          sa..contents = sb..contents & sa..csize = sb..csize"
1199 ensures "True" */
1200 {
1201     /*: assume "0 <= i1 & i1 < sa..csize" */
1202     Object r1a = sa.get(i1);
1203     /*: assume "0 <= i2 & i2 < sa..csize" */
1204     Object r2a = sa.get(i2);
1205     /*: assume "True" */
1206
1207     Object r2b = sb.get(i2);
1208     Object r1b = sb.get(i1);
1209
1210     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1211          sb..csize" */
1212 }
1213
1214 static void get_get_post_c_32(ArrayList sa, ArrayList sb, int i1, int i2)
1215 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1216          sa..contents = sb..contents & sa..csize = sb..csize"
1217 ensures "True" */
1218 {
1219     /*: assume "0 <= i1 & i1 < sa..csize" */
1220     Object r1a = sa.get(i1);
1221     /*: assume "0 <= i2 & i2 < sa..csize" */
1222     Object r2a = sa.get(i2);
1223     /*: assume "~(True)" */
1224
1225     /*: assume "0 <= i2 & i2 < sb..csize" */
1226     Object r2b = sb.get(i2);
1227     /*: assume "0 <= i1 & i1 < sb..csize" */
1228     Object r1b = sb.get(i1);
1229
1230     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1231          sb..csize)" */
1232 }
1233
1234 static void get_indexOf_pre_s_33(ArrayList sa, ArrayList sb, int i1, Object v2)
1235 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1236          sa..contents = sb..contents & sa..csize = sb..csize"
1237 ensures "True" */
1238 {
1239     /*: assume "True" */

```

```

1237     /*: assume "0 <= i1 & i1 < sa..csize" */
1238     Object r1a = sa.get(i1);
1239     int r2a = sa.indexOf(v2);
1240
1241     int r2b = sb.indexOf(v2);
1242     Object r1b = sb.get(i1);
1243
1244     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1245         sb..csize" */
1246 }
1247
1248 static void get_indexOf_pre_c_33(ArrayList sa, ArrayList sb, int i1, Object v2)
1249 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1250     sa..contents = sb..contents & sa..csize = sb..csize"
1251     ensures "True" */
1252 {
1253     /*: assume "~(True)" */
1254     /*: assume "0 <= i1 & i1 < sa..csize" */
1255     Object r1a = sa.get(i1);
1256     int r2a = sa.indexOf(v2);
1257
1258     int r2b = sb.indexOf(v2);
1259     /*: assume "0 <= i1 & i1 < sb..csize" */
1260     Object r1b = sb.get(i1);
1261
1262     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1263         sb..csize)" */
1264 }
1265
1266 static void get_indexOf_between_s_34(ArrayList sa, ArrayList sb, int i1, Object v2)
1267 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1268     sa..contents = sb..contents & sa..csize = sb..csize"
1269     ensures "True" */
1270 {
1271     /*: assume "0 <= i1 & i1 < sa..csize" */
1272     Object r1a = sa.get(i1);
1273     /*: assume "True" */
1274     int r2a = sa.indexOf(v2);
1275
1276     int r2b = sb.indexOf(v2);
1277     Object r1b = sb.get(i1);
1278
1279     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1280         sb..csize" */
1281 }
1282
1283 static void get_indexOf_between_c_34(ArrayList sa, ArrayList sb, int i1, Object v2)
1284 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1285     sa..contents = sb..contents & sa..csize = sb..csize"
1286     ensures "True" */
1287 {
1288     /*: assume "0 <= i1 & i1 < sa..csize" */
1289     Object r1a = sa.get(i1);
1290     /*: assume "~(True)" */
1291     int r2a = sa.indexOf(v2);
1292
1293     int r2b = sb.indexOf(v2);
1294     /*: assume "0 <= i1 & i1 < sb..csize" */
1295     Object r1b = sb.get(i1);
1296
1297     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1298         sb..csize)" */
1299 }
1300
1301 static void get_indexOf_post_s_35(ArrayList sa, ArrayList sb, int i1, Object v2)

```

```

1298  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1299          sa..contents = sb..contents & sa..csize = sb..csize"
1300  ensures "True" */
1301  {
1302    /*: assume "0 <= i1 & i1 < sa..csize" */
1303    Object r1a = sa.get(i1);
1304    int r2a = sa.indexOf(v2);
1305    /*: assume "True" */
1306
1307    int r2b = sb.indexOf(v2);
1308    Object r1b = sb.get(i1);
1309
1310    /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
          sb..csize" */
1311  }
1312
1313  static void get_indexOf_post_c_35(ArrayList sa, ArrayList sb, int i1, Object v2)
1314  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1315          sa..contents = sb..contents & sa..csize = sb..csize"
1316  ensures "True" */
1317  {
1318    /*: assume "0 <= i1 & i1 < sa..csize" */
1319    Object r1a = sa.get(i1);
1320    int r2a = sa.indexOf(v2);
1321    /*: assume "~(True)" */
1322
1323    int r2b = sb.indexOf(v2);
1324    /*: assume "0 <= i1 & i1 < sb..csize" */
1325    Object r1b = sb.get(i1);
1326
1327    /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
          sb..csize)" */
1328  }
1329
1330  static void get_lastIndexOf_pre_s_36(ArrayList sa, ArrayList sb, int i1, Object v2)
1331  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1332          sa..contents = sb..contents & sa..csize = sb..csize"
1333  ensures "True" */
1334  {
1335    /*: assume "True" */
1336    /*: assume "0 <= i1 & i1 < sa..csize" */
1337    Object r1a = sa.get(i1);
1338    int r2a = sa.lastIndexOf(v2);
1339
1340    int r2b = sb.lastIndexOf(v2);
1341    Object r1b = sb.get(i1);
1342
1343    /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
          sb..csize" */
1344  }
1345
1346  static void get_lastIndexOf_pre_c_36(ArrayList sa, ArrayList sb, int i1, Object v2)
1347  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1348          sa..contents = sb..contents & sa..csize = sb..csize"
1349  ensures "True" */
1350  {
1351    /*: assume "~(True)" */
1352    /*: assume "0 <= i1 & i1 < sa..csize" */
1353    Object r1a = sa.get(i1);
1354    int r2a = sa.lastIndexOf(v2);
1355
1356    int r2b = sb.lastIndexOf(v2);
1357    /*: assume "0 <= i1 & i1 < sb..csize" */
1358    Object r1b = sb.get(i1);
1359

```

```

1360     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1361         sb..csize)" */
1362 }
1363 static void get_lastIndexOf_between_s_37(ArrayList sa, ArrayList sb, int i1, Object
1364     v2)
1365 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1366     sa..contents = sb..contents & sa..csize = sb..csize"
1367     ensures "True" */
1368 {
1369     /*: assume "0 <= i1 & i1 < sa..csize" */
1370     Object r1a = sa.get(i1);
1371     /*: assume "True" */
1372     int r2a = sa.lastIndexOf(v2);
1373
1374     int r2b = sb.lastIndexOf(v2);
1375     Object r1b = sb.get(i1);
1376
1377     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1378         sb..csize" */
1379 }
1380 static void get_lastIndexOf_between_c_37(ArrayList sa, ArrayList sb, int i1, Object
1381     v2)
1382 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1383     sa..contents = sb..contents & sa..csize = sb..csize"
1384     ensures "True" */
1385 {
1386     /*: assume "0 <= i1 & i1 < sa..csize" */
1387     Object r1a = sa.get(i1);
1388     /*: assume "~(True)" */
1389     int r2a = sa.lastIndexOf(v2);
1390
1391     int r2b = sb.lastIndexOf(v2);
1392     /*: assume "0 <= i1 & i1 < sb..csize" */
1393     Object r1b = sb.get(i1);
1394
1395     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1396         sb..csize)" */
1397 }
1398 static void get_lastIndexOf_post_s_38(ArrayList sa, ArrayList sb, int i1, Object v2)
1399 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1400     sa..contents = sb..contents & sa..csize = sb..csize"
1401     ensures "True" */
1402 {
1403     /*: assume "0 <= i1 & i1 < sa..csize" */
1404     Object r1a = sa.get(i1);
1405     int r2a = sa.lastIndexOf(v2);
1406     /*: assume "True" */
1407
1408     int r2b = sb.lastIndexOf(v2);
1409     Object r1b = sb.get(i1);
1410
1411     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1412         sb..csize" */
1413 }
1414 static void get_lastIndexOf_post_c_38(ArrayList sa, ArrayList sb, int i1, Object v2)
1415 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1416     sa..contents = sb..contents & sa..csize = sb..csize"
1417     ensures "True" */
1418 {
1419     /*: assume "0 <= i1 & i1 < sa..csize" */
1420     Object r1a = sa.get(i1);

```

```

1419     int r2a = sa.lastIndexOf(v2);
1420     /*: assume "~(True)" */
1421
1422     int r2b = sb.lastIndexOf(v2);
1423     /*: assume "0 <= i1 & i1 < sb..csize" */
1424     Object r1b = sb.get(i1);
1425
1426     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
1427 }
1428
1429 static void get_remove_at_pre_s_39(ArrayList sa, ArrayList sb, int i1, int i2)
1430 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
1431     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1432     ensures "True" */
1433 {
1434     /*: 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)" */
1436     /*: assume "0 <= i1 & i1 < sa..csize" */
1437     Object r1a = sa.get(i1);
1438     /*: assume "0 <= i2 & i2 < sa..csize" */
1439     Object r2a = sa.remove_at(i2);
1440
1441     Object r2b = sb.remove_at(i2);
1442     Object r1b = sb.get(i1);
1443
1444     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
1445 }
1446
1447 static void get_remove_at_pre_c_39(ArrayList sa, ArrayList sb, int i1, int i2)
1448 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
1449     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1450     ensures "True" */
1451 {
1452     /*: 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))" */
1454     /*: assume "0 <= i1 & i1 < sa..csize" */
1455     Object r1a = sa.get(i1);
1456     /*: assume "0 <= i2 & i2 < sa..csize" */
1457     Object r2a = sa.remove_at(i2);
1458
1459     /*: assume "0 <= i2 & i2 < sb..csize" */
1460     Object r2b = sb.remove_at(i2);
1461     /*: assume "0 <= i1 & i1 < sb..csize" */
1462     Object r1b = sb.get(i1);
1463
1464     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
1465 }
1466
1467 static void get_remove_at_between_s_40(ArrayList sa, ArrayList sb, int i1, int i2)
1468 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
1469     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1470     ensures "True" */
1471 {
1472

```

```

1473     /*: assume "0 <= i1 & i1 < sa..csize" */
1474     Object r1a = sa.get(i1);
1475     /*: assume "i1 < i2 | (sa..csize - 1 > i1 & i1 = i2 & (i1 + 1, r1a) :
1476         sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
1477         > i2 & (i1 + 1, r1a) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
1478     /*: assume "0 <= i2 & i2 < sa..csize" */
1479     Object r2a = sa.remove_at(i2);
1480
1481     Object r2b = sb.remove_at(i2);
1482     Object r1b = sb.get(i1);
1483
1484     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1485         sb..csize" */
1486 }
1487
1488 static void get_remove_at_between_c_40(ArrayList sa, ArrayList sb, int i1, int i2)
1489 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1490     sa..contents = sb..contents & sa..csize = sb..csize"
1491 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1492 ensures "True" */
1493 {
1494     /*: assume "0 <= i1 & i1 < sa..csize" */
1495     Object r1a = sa.get(i1);
1496     /*: assume "~(i1 < i2 | (sa..csize - 1 > i1 & i1 = i2 & (i1 + 1, r1a) :
1497         sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
1498         > i2 & (i1 + 1, r1a) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
1499     /*: assume "0 <= i2 & i2 < sa..csize" */
1500     Object r2a = sa.remove_at(i2);
1501
1502     /*: assume "0 <= i2 & i2 < sb..csize" */
1503     Object r2b = sb.remove_at(i2);
1504     /*: assume "0 <= i1 & i1 < sb..csize" */
1505     Object r1b = sb.get(i1);
1506
1507     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1508         sb..csize)" */
1509 }
1510
1511 static void get_remove_at_post_s_41(ArrayList sa, ArrayList sb, int i1, int i2)
1512 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1513     sa..contents = sb..contents & sa..csize = sb..csize"
1514 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1515 ensures "True" */
1516 {
1517     /*: assume "0 <= i1 & i1 < sa..csize" */
1518     Object r1a = sa.get(i1);
1519     /*: assume "0 <= i2 & i2 < sa..csize" */
1520     Object r2a = sa.remove_at(i2);
1521     /*: assume "i1 < i2 | (sa..csize > i1 & i1 = i2 & (i1, r1a) : sa..contents & 0
1522         <= i1 & i1 < sa..csize) | (sa..csize > i1 & i1 > i2 & (i1, r1a) :
1523         sa..contents & 0 <= i1 & i1 < sa..csize)" */
1524
1525     Object r2b = sb.remove_at(i2);
1526     Object r1b = sb.get(i1);
1527
1528     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1529         sb..csize" */
1530 }
1531
1532 static void get_remove_at_post_c_41(ArrayList sa, ArrayList sb, int i1, int i2)
1533 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1534     sa..contents = sb..contents & sa..csize = sb..csize"
1535 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1536 ensures "True" */
1537 {

```



```

1529     /*: assume "0 <= i1 & i1 < sa..csize" */
1530     Object r1a = sa.get(i1);
1531     /*: assume "0 <= i2 & i2 < sa..csize" */
1532     Object r2a = sa.remove_at(i2);
1533     /*: 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))" */
1534
1535     /*: assume "0 <= i2 & i2 < sb..csize" */
1536     Object r2b = sb.remove_at(i2);
1537     /*: assume "0 <= i1 & i1 < sb..csize" */
1538     Object r1b = sb.get(i1);
1539
1540     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
1541 }
1542
1543 static void get_remove_at_pre_s_42(ArrayList sa, ArrayList sb, int i1, int i2)
1544 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1545     sa..contents = sb..contents & sa..csize = sb..csize"
1546     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1547     ensures "True" */
1548 {
1549     /*: 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)" */
1550     /*: assume "0 <= i1 & i1 < sa..csize" */
1551     Object r1a = sa.get(i1);
1552     /*: assume "0 <= i2 & i2 < sa..csize" */
1553     sa.remove_at(i2);
1554
1555     sb.remove_at(i2);
1556     Object r1b = sb.get(i1);
1557
1558     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
1559 }
1560
1561 static void get_remove_at_pre_c_42(ArrayList sa, ArrayList sb, int i1, int i2)
1562 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1563     sa..contents = sb..contents & sa..csize = sb..csize"
1564     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1565     ensures "True" */
1566 {
1567     /*: 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))" */
1568     /*: assume "0 <= i1 & i1 < sa..csize" */
1569     Object r1a = sa.get(i1);
1570     /*: assume "0 <= i2 & i2 < sa..csize" */
1571     sa.remove_at(i2);
1572
1573     /*: assume "0 <= i2 & i2 < sb..csize" */
1574     sb.remove_at(i2);
1575     /*: assume "0 <= i1 & i1 < sb..csize" */
1576     Object r1b = sb.get(i1);
1577
1578     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
1579 }
1580
1581 static void get_remove_at_between_s_43(ArrayList sa, ArrayList sb, int i1, int i2)

```

```

1582  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1583             sa..contents = sb..contents & sa..csize = sb..csize"
1584  modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1585  ensures "True" */
1586  {
1587      /*: assume "0 <= i1 & i1 < sa..csize" */
1588      Object r1a = sa.get(i1);
1589      /*: assume "i1 < i2 | (sa..csize - 1 > i1 & i1 = i2 & (i1 + 1, r1a) :
1590             sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
1591             > i2 & (i1 + 1, r1a) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
1592      /*: assume "0 <= i2 & i2 < sa..csize" */
1593      sa.remove_at(i2);
1594
1595      sb.remove_at(i2);
1596      Object r1b = sb.get(i1);
1597
1598      /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
1599  }
1600
1601  static void get_remove_at_between_c_43(ArrayList sa, ArrayList sb, int i1, int i2)
1602  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1603             sa..contents = sb..contents & sa..csize = sb..csize"
1604  modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1605  ensures "True" */
1606  {
1607      /*: assume "0 <= i1 & i1 < sa..csize" */
1608      Object r1a = sa.get(i1);
1609      /*: assume "~(i1 < i2 | (sa..csize - 1 > i1 & i1 = i2 & (i1 + 1, r1a) :
1610             sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
1611             > i2 & (i1 + 1, r1a) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
1612      /*: assume "0 <= i2 & i2 < sa..csize" */
1613      sa.remove_at(i2);
1614
1615      /*: assume "0 <= i2 & i2 < sb..csize" */
1616      sb.remove_at(i2);
1617      /*: assume "0 <= i1 & i1 < sb..csize" */
1618      Object r1b = sb.get(i1);
1619
1620      /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
1621             */
1622  }
1623
1624  static void get_remove_at_post_s_44(ArrayList sa, ArrayList sb, int i1, int i2)
1625  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1626             sa..contents = sb..contents & sa..csize = sb..csize"
1627  modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1628  ensures "True" */
1629  {
1630      /*: assume "0 <= i1 & i1 < sa..csize" */
1631      Object r1a = sa.get(i1);
1632      /*: assume "0 <= i2 & i2 < sa..csize" */
1633      sa.remove_at(i2);
1634      /*: assume "i1 < i2 | (sa..csize > i1 & i1 = i2 & (i1, r1a) : sa..contents & 0
1635             <= i1 & i1 < sa..csize) | (sa..csize > i1 & i1 > i2 & (i1, r1a) :
1636             sa..contents & 0 <= i1 & i1 < sa..csize)" */
1637
1638      sb.remove_at(i2);
1639      Object r1b = sb.get(i1);
1640
1641      /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
1642  }
1643
1644  static void get_remove_at_post_c_44(ArrayList sa, ArrayList sb, int i1, int i2)
1645  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1646             sa..contents = sb..contents & sa..csize = sb..csize"

```

```

1640     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1641     ensures "True" */
1642 {
1643     /*: assume "0 <= i1 & i1 < sa..csize" */
1644     Object r1a = sa.get(i1);
1645     /*: assume "0 <= i2 & i2 < sa..csize" */
1646     sa.remove_at(i2);
1647     /*: assume "~(i1 < i2 | (sa..csize > i1 & i1 = i2 & (i1, r1a) : sa..contents &
1648         0 <= i1 & i1 < sa..csize) | (sa..csize > i1 & i1 > i2 & (i1, r1a) :
1649         sa..contents & 0 <= i1 & i1 < sa..csize))" */
1650     /*: assume "0 <= i2 & i2 < sb..csize" */
1651     sb.remove_at(i2);
1652     /*: assume "0 <= i1 & i1 < sb..csize" */
1653     Object r1b = sb.get(i1);
1654     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
1655         */
1656 }
1657
1658 static void get_set_pre_s_45(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
1659 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1660     sa..contents = sb..contents & sa..csize = sb..csize"
1661     modifies "sa..contents", "sb..contents"
1662     ensures "True" */
1663 {
1664     /*: assume "i1 < i2 | (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 <
1665         sa..csize) | i1 > i2" */
1666     /*: assume "0 <= i1 & i1 < sa..csize" */
1667     Object r1a = sa.get(i1);
1668     /*: assume "0 <= i2 & i2 < sa..csize" */
1669     Object r2a = sa.set(i2, v2);
1670     Object r2b = sb.set(i2, v2);
1671     Object r1b = sb.get(i1);
1672     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1673         sb..csize" */
1674 }
1675
1676 static void get_set_pre_c_45(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
1677 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1678     sa..contents = sb..contents & sa..csize = sb..csize"
1679     modifies "sa..contents", "sb..contents"
1680     ensures "True" */
1681 {
1682     /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 <
1683         sa..csize) | i1 > i2)" */
1684     /*: assume "0 <= i1 & i1 < sa..csize" */
1685     Object r1a = sa.get(i1);
1686     /*: assume "0 <= i2 & i2 < sa..csize" */
1687     Object r2a = sa.set(i2, v2);
1688     /*: assume "0 <= i2 & i2 < sb..csize" */
1689     Object r2b = sb.set(i2, v2);
1690     /*: assume "0 <= i1 & i1 < sb..csize" */
1691     Object r1b = sb.get(i1);
1692     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1693         sb..csize)" */
1694 }
1695
1696 static void get_set_between_s_46(ArrayList sa, ArrayList sb, int i1, int i2, Object
1697     v2)
1698 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &

```

```

1697         sa..contents = sb..contents & sa..csize = sb..csize"
1698     modifies "sa..contents", "sb..contents"
1699     ensures "True" */
1700 {
1701     /*: assume "0 <= i1 & i1 < sa..csize" */
1702     Object r1a = sa.get(i1);
1703     /*: assume "i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2" */
1704     /*: assume "0 <= i2 & i2 < sa..csize" */
1705     Object r2a = sa.set(i2, v2);
1706
1707     Object r2b = sb.set(i2, v2);
1708     Object r1b = sb.get(i1);
1709
1710     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1711         sb..csize" */
1712 }
1713
1714 static void get_set_between_c_46(ArrayList sa, ArrayList sb, int i1, int i2, Object
1715     v2)
1716 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1717     sa..contents = sb..contents & sa..csize = sb..csize"
1718     modifies "sa..contents", "sb..contents"
1719     ensures "True" */
1720 {
1721     /*: assume "0 <= i1 & i1 < sa..csize" */
1722     Object r1a = sa.get(i1);
1723     /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2)" */
1724     /*: assume "0 <= i2 & i2 < sa..csize" */
1725     Object r2a = sa.set(i2, v2);
1726
1727     /*: assume "0 <= i2 & i2 < sb..csize" */
1728     Object r2b = sb.set(i2, v2);
1729     /*: assume "0 <= i1 & i1 < sb..csize" */
1730     Object r1b = sb.get(i1);
1731
1732     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1733         sb..csize)" */
1734 }
1735
1736 static void get_set_post_s_47(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
1737 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1738     sa..contents = sb..contents & sa..csize = sb..csize"
1739     modifies "sa..contents", "sb..contents"
1740     ensures "True" */
1741 {
1742     /*: assume "0 <= i1 & i1 < sa..csize" */
1743     Object r1a = sa.get(i1);
1744     /*: assume "0 <= i2 & i2 < sa..csize" */
1745     Object r2a = sa.set(i2, v2);
1746     /*: assume "i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2" */
1747
1748     Object r2b = sb.set(i2, v2);
1749     Object r1b = sb.get(i1);
1750
1751     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1752         sb..csize" */
1753 }
1754
1755 static void get_set_post_c_47(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
1756 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1757     sa..contents = sb..contents & sa..csize = sb..csize"
1758     modifies "sa..contents", "sb..contents"
1759     ensures "True" */
1760 {
1761     /*: assume "0 <= i1 & i1 < sa..csize" */

```

```

1758     Object r1a = sa.get(i1);
1759     /*: assume "0 <= i2 & i2 < sa..csize" */
1760     Object r2a = sa.set(i2, v2);
1761     /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2)" */
1762
1763     /*: assume "0 <= i2 & i2 < sb..csize" */
1764     Object r2b = sb.set(i2, v2);
1765     /*: assume "0 <= i1 & i1 < sb..csize" */
1766     Object r1b = sb.get(i1);
1767
1768     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
1769 }
1770
1771 static void get_set_pre_s_48(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
1772 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1773     sa..contents = sb..contents & sa..csize = sb..csize"
1774     modifies "sa..contents", "sb..contents"
1775     ensures "True" */
1776 {
1777     /*: assume "i1 < i2 | (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 <
        sa..csize) | i1 > i2" */
1778     /*: assume "0 <= i1 & i1 < sa..csize" */
1779     Object r1a = sa.get(i1);
1780     /*: assume "0 <= i2 & i2 < sa..csize" */
1781     sa.set(i2, v2);
1782
1783     sb.set(i2, v2);
1784     Object r1b = sb.get(i1);
1785
1786     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
1787 }
1788
1789 static void get_set_pre_c_48(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
1790 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1791     sa..contents = sb..contents & sa..csize = sb..csize"
1792     modifies "sa..contents", "sb..contents"
1793     ensures "True" */
1794 {
1795     /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 <
        sa..csize) | i1 > i2)" */
1796     /*: assume "0 <= i1 & i1 < sa..csize" */
1797     Object r1a = sa.get(i1);
1798     /*: assume "0 <= i2 & i2 < sa..csize" */
1799     sa.set(i2, v2);
1800
1801     /*: assume "0 <= i2 & i2 < sb..csize" */
1802     sb.set(i2, v2);
1803     /*: assume "0 <= i1 & i1 < sb..csize" */
1804     Object r1b = sb.get(i1);
1805
1806     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
1807 }
1808
1809 static void get_set_between_s_49(ArrayList sa, ArrayList sb, int i1, int i2, Object
v2)
1810 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1811     sa..contents = sb..contents & sa..csize = sb..csize"
1812     modifies "sa..contents", "sb..contents"
1813     ensures "True" */
1814 {
1815     /*: assume "0 <= i1 & i1 < sa..csize" */
1816     Object r1a = sa.get(i1);
1817     /*: assume "i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2" */

```

```

1818     /*: assume "0 <= i2 & i2 < sa..csize" */
1819     sa.set(i2, v2);
1820
1821     sb.set(i2, v2);
1822     Object r1b = sb.get(i1);
1823
1824     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
1825 }
1826
1827 static void get_set_between_c_49(ArrayList sa, ArrayList sb, int i1, int i2, Object
1828     v2)
1829 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1830     sa..contents = sb..contents & sa..csize = sb..csize"
1831     modifies "sa..contents", "sb..contents"
1832     ensures "True" */
1833 {
1834     /*: assume "0 <= i1 & i1 < sa..csize" */
1835     Object r1a = sa.get(i1);
1836     /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2)" */
1837     /*: assume "0 <= i2 & i2 < sa..csize" */
1838     sa.set(i2, v2);
1839
1840     /*: assume "0 <= i2 & i2 < sb..csize" */
1841     sb.set(i2, v2);
1842     /*: assume "0 <= i1 & i1 < sb..csize" */
1843     Object r1b = sb.get(i1);
1844
1845     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
1846         */
1847 }
1848
1849 static void get_set_post_s_50(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
1850 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1851     sa..contents = sb..contents & sa..csize = sb..csize"
1852     modifies "sa..contents", "sb..contents"
1853     ensures "True" */
1854 {
1855     /*: assume "0 <= i1 & i1 < sa..csize" */
1856     Object r1a = sa.get(i1);
1857     /*: assume "0 <= i2 & i2 < sa..csize" */
1858     sa.set(i2, v2);
1859     /*: assume "i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2" */
1860
1861     sb.set(i2, v2);
1862     Object r1b = sb.get(i1);
1863
1864     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
1865 }
1866
1867 static void get_set_post_c_50(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
1868 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1869     sa..contents = sb..contents & sa..csize = sb..csize"
1870     modifies "sa..contents", "sb..contents"
1871     ensures "True" */
1872 {
1873     /*: assume "0 <= i1 & i1 < sa..csize" */
1874     Object r1a = sa.get(i1);
1875     /*: assume "0 <= i2 & i2 < sa..csize" */
1876     sa.set(i2, v2);
1877     /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2)" */
1878
1879     /*: assume "0 <= i2 & i2 < sb..csize" */
1880     sb.set(i2, v2);
1881     /*: assume "0 <= i1 & i1 < sb..csize" */
1882     Object r1b = sb.get(i1);

```

```

1881     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
1882     */
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 =
1899     sb..csize" */
1900 }
1901
1902 static void get_size_pre_c_51(ArrayList sa, ArrayList sb, int i1)
1903 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1904     sa..contents = sb..contents & sa..csize = sb..csize"
1905     ensures "True" */
1906 {
1907     /*: assume "~(True)" */
1908     /*: assume "0 <= i1 & i1 < sa..csize" */
1909     Object r1a = sa.get(i1);
1910     int r2a = sa.size();
1911
1912     int r2b = sb.size();
1913     /*: assume "0 <= i1 & i1 < sb..csize" */
1914     Object r1b = sb.get(i1);
1915
1916     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1917     sb..csize)" */
1918 }
1919
1920 static void get_size_between_s_52(ArrayList sa, ArrayList sb, int i1)
1921 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1922     sa..contents = sb..contents & sa..csize = sb..csize"
1923     ensures "True" */
1924 {
1925     /*: assume "0 <= i1 & i1 < sa..csize" */
1926     Object r1a = sa.get(i1);
1927     /*: assume "True" */
1928     int r2a = sa.size();
1929
1930     int r2b = sb.size();
1931     Object r1b = sb.get(i1);
1932
1933     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1934     sb..csize" */
1935 }
1936
1937 static void get_size_between_c_52(ArrayList sa, ArrayList sb, int i1)
1938 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1939     sa..contents = sb..contents & sa..csize = sb..csize"
1940     ensures "True" */
1941 {
1942     /*: assume "0 <= i1 & i1 < sa..csize" */
1943     Object r1a = sa.get(i1);
1944     /*: 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" */

```



```

1999 }
2000
2001 static void indexOf_add_at_pre_c_54(ArrayList sa, ArrayList sb, Object v1, int i2,
2002     Object v2)
2003 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2004     sa..contents = sb..contents & sa..csize = sb..csize"
2005     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
2006     "sb..msize"
2007     ensures "True" */
2008 {
2009     /*: assume "~((~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
2010     v2) | (EX i. (i, v1) : sa..contents & 0 <= i & i < i2) | ~(EX i. (i, v1) :
2011     sa..contents & 0 <= i & i < i2) & (i2, v1) : sa..contents & 0 <= i2 & i2 <
2012     sa..csize & v1 = v2))" */
2013     int r1a = sa.indexOf(v1);
2014     /*: assume "0 <= i2 & i2 <= sa..csize" */
2015     sa.add_at(i2, v2);
2016
2017     /*: assume "0 <= i2 & i2 <= sb..csize" */
2018     sb.add_at(i2, v2);
2019     int r1b = sb.indexOf(v1);
2020
2021     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
2022     */
2023 }
2024
2025 static void indexOf_add_at_between_s_55(ArrayList sa, ArrayList sb, Object v1, int
2026     i2, Object v2)
2027 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2028     sa..contents = sb..contents & sa..csize = sb..csize"
2029     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
2030     "sb..msize"
2031     ensures "True" */
2032 {
2033     int r1a = sa.indexOf(v1);
2034     /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 =
2035     v2)" */
2036     /*: assume "0 <= i2 & i2 <= sa..csize" */
2037     sa.add_at(i2, v2);
2038
2039     sb.add_at(i2, v2);
2040     int r1b = sb.indexOf(v1);
2041
2042     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
2043 }
2044
2045 static void indexOf_add_at_between_c_55(ArrayList sa, ArrayList sb, Object v1, int
2046     i2, Object v2)
2047 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2048     sa..contents = sb..contents & sa..csize = sb..csize"
2049     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
2050     "sb..msize"
2051     ensures "True" */
2052 {
2053     int r1a = sa.indexOf(v1);
2054     /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 =
2055     v2))" */
2056     /*: assume "0 <= i2 & i2 <= sa..csize" */
2057     sa.add_at(i2, v2);
2058
2059     /*: assume "0 <= i2 & i2 <= sb..csize" */
2060     sb.add_at(i2, v2);
2061     int r1b = sb.indexOf(v1);

```

```

2051     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
2052         */
2053 }
2054 static void index0f_add_at_post_s_56(ArrayList sa, ArrayList sb, Object v1, int i2,
2055     Object v2)
2056 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2057     sa..contents = sb..contents & sa..csize = sb..csize"
2058     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
2059     "sb..msize"
2060     ensures "True" */
2061 {
2062     int r1a = sa.index0f(v1);
2063     /*: assume "0 <= i2 & i2 <= sa..csize" */
2064     sa.add_at(i2, v2);
2065     /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 =
2066         v2)" */
2067     sb.add_at(i2, v2);
2068     int r1b = sb.index0f(v1);
2069     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
2070 }
2071 static void index0f_add_at_post_c_56(ArrayList sa, ArrayList sb, Object v1, int i2,
2072     Object v2)
2073 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2074     sa..contents = sb..contents & sa..csize = sb..csize"
2075     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
2076     "sb..msize"
2077     ensures "True" */
2078 {
2079     int r1a = sa.index0f(v1);
2080     /*: assume "0 <= i2 & i2 <= sa..csize" */
2081     sa.add_at(i2, v2);
2082     /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 =
2083         v2))" */
2084     sb.add_at(i2, v2);
2085     int r1b = sb.index0f(v1);
2086     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
2087         */
2088 }
2089 static void index0f_get_pre_s_57(ArrayList sa, ArrayList sb, Object v1, int i2)
2090 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2091     sa..contents = sb..contents & sa..csize = sb..csize"
2092     ensures "True" */
2093 {
2094     /*: assume "True" */
2095     int r1a = sa.index0f(v1);
2096     /*: assume "0 <= i2 & i2 < sa..csize" */
2097     Object r2a = sa.get(i2);
2098
2099     Object r2b = sb.get(i2);
2100     int r1b = sb.index0f(v1);
2101
2102     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2103         sb..csize" */
2104 }
2105 static void index0f_get_pre_c_57(ArrayList sa, ArrayList sb, Object v1, int i2)
2106 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &

```

```

2107         sa..contents = sb..contents & sa..csize = sb..csize"
2108     ensures "True" */
2109 {
2110     /*: assume "~(True)" */
2111     int r1a = sa.indexOf(v1);
2112     /*: assume "0 <= i2 & i2 < sa..csize" */
2113     Object r2a = sa.get(i2);
2114
2115     /*: assume "0 <= i2 & i2 < sb..csize" */
2116     Object r2b = sb.get(i2);
2117     int r1b = sb.indexOf(v1);
2118
2119     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2120         sb..csize)" */
2121 }
2122
2123 static void indexOf_get_between_s_58(ArrayList sa, ArrayList sb, Object v1, int i2)
2124 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2125     sa..contents = sb..contents & sa..csize = sb..csize"
2126     ensures "True" */
2127 {
2128     int r1a = sa.indexOf(v1);
2129     /*: assume "True" */
2130     /*: assume "0 <= i2 & i2 < sa..csize" */
2131     Object r2a = sa.get(i2);
2132
2133     Object r2b = sb.get(i2);
2134     int r1b = sb.indexOf(v1);
2135
2136     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2137         sb..csize" */
2138 }
2139
2140 static void indexOf_get_between_c_58(ArrayList sa, ArrayList sb, Object v1, int i2)
2141 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2142     sa..contents = sb..contents & sa..csize = sb..csize"
2143     ensures "True" */
2144 {
2145     int r1a = sa.indexOf(v1);
2146     /*: assume "~(True)" */
2147     /*: assume "0 <= i2 & i2 < sa..csize" */
2148     Object r2a = sa.get(i2);
2149
2150     /*: assume "0 <= i2 & i2 < sb..csize" */
2151     Object r2b = sb.get(i2);
2152     int r1b = sb.indexOf(v1);
2153
2154     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2155         sb..csize)" */
2156 }
2157
2158 static void indexOf_get_post_s_59(ArrayList sa, ArrayList sb, Object v1, int i2)
2159 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2160     sa..contents = sb..contents & sa..csize = sb..csize"
2161     ensures "True" */
2162 {
2163     int r1a = sa.indexOf(v1);
2164     /*: assume "0 <= i2 & i2 < sa..csize" */
2165     Object r2a = sa.get(i2);
2166     /*: assume "True" */
2167
2168     Object r2b = sb.get(i2);
2169     int r1b = sb.indexOf(v1);

```

```

2168     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2169         sb..csize" */
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 =
2186         sb..csize)" */
2187 }
2188 static void indexOf_indexOf_pre_s_60(ArrayList sa, ArrayList sb, Object v1, Object
2189     v2)
2190 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2191     sa..contents = sb..contents & sa..csize = sb..csize"
2192     ensures "True" */
2193 {
2194     /*: assume "True" */
2195     int r1a = sa.indexOf(v1);
2196     int r2a = sa.indexOf(v2);
2197
2198     int r2b = sb.indexOf(v2);
2199     int r1b = sb.indexOf(v1);
2200
2201     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2202         sb..csize" */
2203 }
2204 static void indexOf_indexOf_pre_c_60(ArrayList sa, ArrayList sb, Object v1, Object
2205     v2)
2206 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2207     sa..contents = sb..contents & sa..csize = sb..csize"
2208     ensures "True" */
2209 {
2210     /*: assume "~(True)" */
2211     int r1a = sa.indexOf(v1);
2212     int r2a = sa.indexOf(v2);
2213
2214     int r2b = sb.indexOf(v2);
2215     int r1b = sb.indexOf(v1);
2216
2217     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2218         sb..csize)" */
2219 }
2220 static void indexOf_indexOf_between_s_61(ArrayList sa, ArrayList sb, Object v1,
2221     Object v2)
2222 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2223     sa..contents = sb..contents & sa..csize = sb..csize"
2224     ensures "True" */
2225 {
2226     int r1a = sa.indexOf(v1);
2227     /*: assume "True" */
2228     int r2a = sa.indexOf(v2);

```

```

2226     int r2b = sb.indexOf(v2);
2227     int r1b = sb.indexOf(v1);
2228
2229     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2230         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 =
2246         sb..csize)" */
2247 }
2248
2249 static void indexOf_indexOf_post_s_62(ArrayList sa, ArrayList sb, Object v1, Object
    v2)
2250 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2251     sa..contents = sb..contents & sa..csize = sb..csize"
2252     ensures "True" */
2253 {
2254     int r1a = sa.indexOf(v1);
2255     int r2a = sa.indexOf(v2);
2256     /*: assume "True" */
2257
2258     int r2b = sb.indexOf(v2);
2259     int r1b = sb.indexOf(v1);
2260
2261     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2262         sb..csize" */
2263 }
2264
2265 static void indexOf_indexOf_post_c_62(ArrayList sa, ArrayList sb, Object v1, Object
    v2)
2266 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2267     sa..contents = sb..contents & sa..csize = sb..csize"
2268     ensures "True" */
2269 {
2270     int r1a = sa.indexOf(v1);
2271     int r2a = sa.indexOf(v2);
2272     /*: assume "~(True)" */
2273
2274     int r2b = sb.indexOf(v2);
2275     int r1b = sb.indexOf(v1);
2276
2277     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2278         sb..csize)" */
2279 }
2280
2281 static void indexOf_lastIndexOf_pre_s_63(ArrayList sa, ArrayList sb, Object v1,
    Object v2)
2282 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2283     sa..contents = sb..contents & sa..csize = sb..csize"
2284     ensures "True" */
2285 {

```

```

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 =
2291         sb..csize" */
2292 }
2293
2294 static void indexOf_lastIndexOf_pre_c_63(ArrayList sa, ArrayList sb, Object v1,
2295     Object v2)
2296 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2297     sa..contents = sb..contents & sa..csize = sb..csize"
2298     ensures "True" */
2299 {
2300     /*: assume "~(True)" */
2301     int r1a = sa.indexOf(v1);
2302     int r2a = sa.lastIndexOf(v2);
2303
2304     int r2b = sb.lastIndexOf(v2);
2305     int r1b = sb.indexOf(v1);
2306
2307     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2308         sb..csize)" */
2309 }
2310
2311 static void indexOf_lastIndexOf_between_s_64(ArrayList sa, ArrayList sb, Object v1,
2312     Object v2)
2313 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2314     sa..contents = sb..contents & sa..csize = sb..csize"
2315     ensures "True" */
2316 {
2317     int r1a = sa.indexOf(v1);
2318     /*: assume "True" */
2319     int r2a = sa.lastIndexOf(v2);
2320
2321     int r2b = sb.lastIndexOf(v2);
2322     int r1b = sb.indexOf(v1);
2323
2324     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2325         sb..csize" */
2326 }
2327
2328 static void indexOf_lastIndexOf_between_c_64(ArrayList sa, ArrayList sb, Object v1,
2329     Object v2)
2330 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2331     sa..contents = sb..contents & sa..csize = sb..csize"
2332     ensures "True" */
2333 {
2334     int r1a = sa.indexOf(v1);
2335     /*: assume "~(True)" */
2336     int r2a = sa.lastIndexOf(v2);
2337
2338     int r2b = sb.lastIndexOf(v2);
2339     int r1b = sb.indexOf(v1);
2340
2341     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2342         sb..csize)" */
2343 }
2344
2345 static void indexOf_lastIndexOf_post_s_65(ArrayList sa, ArrayList sb, Object v1,
2346     Object v2)
2347 /*: 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 &
        sa..contents = sb..contents & sa..csize = sb..csize"
2355     ensures "True" */
2356 {
2357     int r1a = sa.indexOf(v1);
2358     int r2a = sa.lastIndexOf(v2);
2359     /*: assume "~(True)" */
2360
2361     int r2b = sb.lastIndexOf(v2);
2362     int r1b = sb.indexOf(v1);
2363
2364     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
2365 }
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 &
        sa..contents = sb..contents & sa..csize = sb..csize"
2370     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2371     ensures "True" */
2372 {
2373     /*: 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)" */
2374     int r1a = sa.indexOf(v1);
2375     /*: assume "0 <= i2 & i2 < sa..csize" */
2376     Object r2a = sa.remove_at(i2);
2377
2378     Object r2b = sb.remove_at(i2);
2379     int r1b = sb.indexOf(v1);
2380
2381     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
2382 }
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 &
        sa..contents = sb..contents & sa..csize = sb..csize"
2387     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2388     ensures "True" */
2389 {
2390     /*: 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

```

```

2392         < sa..csize - 1 & (i2 + 1, v1) : sa..contents & 0 <= i2 + 1 & i2 + 1 <
2393         sa..csize))" */
2394     int r1a = sa.indexOf(v1);
2395     /*: assume "0 <= i2 & i2 < sa..csize" */
2396     Object r2a = sa.remove_at(i2);
2397
2398     /*: assume "0 <= i2 & i2 < sb..csize" */
2399     Object r2b = sb.remove_at(i2);
2400     int r1b = sb.indexOf(v1);
2401
2402     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2403     sb..csize)" */
2404 }
2405
2406 static void indexOf_remove_at_between_s_67(ArrayList sa, ArrayList sb, Object v1,
2407     int i2)
2408 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2409     sa..contents = sb..contents & sa..csize = sb..csize"
2410 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2411 ensures "True" */
2412 {
2413     int r1a = sa.indexOf(v1);
2414     /*: assume "r1a < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize - 1 &
2415     (i2 + 1, v1) : sa..contents & 0 <= i2 + 1 & i2 + 1 < sa..csize)" */
2416     /*: assume "0 <= i2 & i2 < sa..csize" */
2417     Object r2a = sa.remove_at(i2);
2418
2419     Object r2b = sb.remove_at(i2);
2420     int r1b = sb.indexOf(v1);
2421
2422     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2423     sb..csize" */
2424 }
2425
2426 static void indexOf_remove_at_between_c_67(ArrayList sa, ArrayList sb, Object v1,
2427     int i2)
2428 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2429     sa..contents = sb..contents & sa..csize = sb..csize"
2430 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2431 ensures "True" */
2432 {
2433     int r1a = sa.indexOf(v1);
2434     /*: assume "~(r1a < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize - 1
2435     & (i2 + 1, v1) : sa..contents & 0 <= i2 + 1 & i2 + 1 < sa..csize))" */
2436     /*: assume "0 <= i2 & i2 < sa..csize" */
2437     Object r2a = sa.remove_at(i2);
2438
2439     /*: assume "0 <= i2 & i2 < sb..csize" */
2440     Object r2b = sb.remove_at(i2);
2441     int r1b = sb.indexOf(v1);
2442
2443     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2444     sb..csize)" */
2445 }
2446
2447 static void indexOf_remove_at_post_s_68(ArrayList sa, ArrayList sb, Object v1, int
2448     i2)
2449 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2450     sa..contents = sb..contents & sa..csize = sb..csize"
2451 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2452 ensures "True" */
2453 {
2454     int r1a = sa.indexOf(v1);
2455     /*: assume "0 <= i2 & i2 < sa..csize" */
2456     Object r2a = sa.remove_at(i2);

```



```

2447     /*: assume "r1a < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize & (i2,
2448         v1) : sa..contents & 0 <= i2 & i2 < sa..csize)" */
2449
2450     Object r2b = sb.remove_at(i2);
2451     int r1b = sb.indexOf(v1);
2452
2453     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2454         sb..csize" */
2455 }
2456
2457 static void indexOf_remove_at_post_c_68(ArrayList sa, ArrayList sb, Object v1, int
2458     i2)
2459 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2460     sa..contents = sb..contents & sa..csize = sb..csize"
2461     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2462     ensures "True" */
2463 {
2464     int r1a = sa.indexOf(v1);
2465     /*: assume "0 <= i2 & i2 < sa..csize" */
2466     Object r2a = sa.remove_at(i2);
2467     /*: assume "~(r1a < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize &
2468         (i2, v1) : sa..contents & 0 <= i2 & i2 < sa..csize))" */
2469
2470     /*: assume "0 <= i2 & i2 < sb..csize" */
2471     Object r2b = sb.remove_at(i2);
2472     int r1b = sb.indexOf(v1);
2473
2474     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2475         sb..csize)" */
2476 }
2477
2478 static void indexOf_remove_at_pre_s_69(ArrayList sa, ArrayList sb, Object v1, int
2479     i2)
2480 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2481     sa..contents = sb..contents & sa..csize = sb..csize"
2482     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2483     ensures "True" */
2484 {
2485     /*: assume "~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
2486         (i, v1) : sa..contents & 0 <= i & i < i2) | (~(EX i. (i, v1) : sa..contents
2487         & 0 <= i & i < i2) & (i2, v1) : sa..contents & 0 <= i2 & i2 < sa..csize & i2
2488         < sa..csize - 1 & (i2 + 1, v1) : sa..contents & 0 <= i2 + 1 & i2 + 1 <
2489         sa..csize))" */
2490     int r1a = sa.indexOf(v1);
2491     /*: assume "0 <= i2 & i2 < sa..csize" */
2492     sa.remove_at(i2);
2493
2494     sb.remove_at(i2);
2495     int r1b = sb.indexOf(v1);
2496
2497     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
2498 }
2499
2500 static void indexOf_remove_at_pre_c_69(ArrayList sa, ArrayList sb, Object v1, int
2501     i2)
2502 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2503     sa..contents = sb..contents & sa..csize = sb..csize"
2504     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2505     ensures "True" */
2506 {
2507     /*: assume "~(~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
2508         (i, v1) : sa..contents & 0 <= i & i < i2) | (~(EX i. (i, v1) : sa..contents
2509         & 0 <= i & i < i2) & (i2, v1) : sa..contents & 0 <= i2 & i2 < sa..csize & i2
2510         < sa..csize - 1 & (i2 + 1, v1) : sa..contents & 0 <= i2 + 1 & i2 + 1 <
2511         sa..csize))" */

```

```

2497     int r1a = sa.indexOf(v1);
2498     /*: assume "0 <= i2 & i2 < sa..csize" */
2499     sa.remove_at(i2);
2500
2501     /*: assume "0 <= i2 & i2 < sb..csize" */
2502     sb.remove_at(i2);
2503     int r1b = sb.indexOf(v1);
2504
2505     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
2506 }
2507
2508 static void indexOf_remove_at_between_s_70(ArrayList sa, ArrayList sb, Object v1,
        int i2)
2509 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2510             sa..contents = sb..contents & sa..csize = sb..csize"
2511     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2512     ensures "True" */
2513 {
2514     int r1a = sa.indexOf(v1);
2515     /*: 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)" */
2516     /*: assume "0 <= i2 & i2 < sa..csize" */
2517     sa.remove_at(i2);
2518
2519     sb.remove_at(i2);
2520     int r1b = sb.indexOf(v1);
2521
2522     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
2523 }
2524
2525 static void indexOf_remove_at_between_c_70(ArrayList sa, ArrayList sb, Object v1,
        int i2)
2526 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2527             sa..contents = sb..contents & sa..csize = sb..csize"
2528     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2529     ensures "True" */
2530 {
2531     int r1a = sa.indexOf(v1);
2532     /*: 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))" */
2533     /*: assume "0 <= i2 & i2 < sa..csize" */
2534     sa.remove_at(i2);
2535
2536     /*: assume "0 <= i2 & i2 < sb..csize" */
2537     sb.remove_at(i2);
2538     int r1b = sb.indexOf(v1);
2539
2540     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
2541 }
2542
2543 static void indexOf_remove_at_post_s_71(ArrayList sa, ArrayList sb, Object v1, int
        i2)
2544 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2545             sa..contents = sb..contents & sa..csize = sb..csize"
2546     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2547     ensures "True" */
2548 {
2549     int r1a = sa.indexOf(v1);
2550     /*: assume "0 <= i2 & i2 < sa..csize" */
2551     sa.remove_at(i2);
2552     /*: assume "r1a < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize & (i2,
        v1) : sa..contents & 0 <= i2 & i2 < sa..csize)" */
2553

```

```

2554     sb.remove_at(i2);
2555     int r1b = sb.indexOf(v1);
2556
2557     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
2558 }
2559
2560 static void indexOf_remove_at_post_c_71(ArrayList sa, ArrayList sb, Object v1, int
2561     i2)
2562 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2563     sa..contents = sb..contents & sa..csize = sb..csize"
2564 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2565 ensures "True" */
2566 {
2567     int r1a = sa.indexOf(v1);
2568     /*: assume "0 <= i2 & i2 < sa..csize" */
2569     sa.remove_at(i2);
2570     /*: assume "~(r1a < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize &
2571     (i2, v1) : sa..contents & 0 <= i2 & i2 < sa..csize))" */
2572
2573     /*: assume "0 <= i2 & i2 < sb..csize" */
2574     sb.remove_at(i2);
2575     int r1b = sb.indexOf(v1);
2576
2577     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
2578     */
2579 }
2580
2581 static void indexOf_set_pre_s_72(ArrayList sa, ArrayList sb, Object v1, int i2,
2582     Object v2)
2583 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2584     sa..contents = sb..contents & sa..csize = sb..csize"
2585 modifies "sa..contents", "sb..contents"
2586 ensures "True" */
2587 {
2588     /*: assume "~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
2589     v2) | (EX i. (i, v1) : sa..contents & 0 <= i & i < i2) | ~(EX i. (i, v1) :
2590     sa..contents & 0 <= i & i < i2) & (i2, v1) : sa..contents & v1 = v2) | ~(EX
2591     i. (i, v1) : sa..contents & 0 <= i & i <= i2) & (EX i. (i, v1) :
2592     sa..contents & i2 < i & i < sa..csize) & v1 ~= v2)" */
2593     int r1a = sa.indexOf(v1);
2594     /*: assume "0 <= i2 & i2 < sa..csize" */
2595     Object r2a = sa.set(i2, v2);
2596
2597     Object r2b = sb.set(i2, v2);
2598     int r1b = sb.indexOf(v1);
2599
2600     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2601     sb..csize" */
2602 }
2603
2604 static void indexOf_set_pre_c_72(ArrayList sa, ArrayList sb, Object v1, int i2,
2605     Object v2)
2606 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2607     sa..contents = sb..contents & sa..csize = sb..csize"
2608 modifies "sa..contents", "sb..contents"
2609 ensures "True" */
2610 {
2611     /*: assume "~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
2612     v2) | (EX i. (i, v1) : sa..contents & 0 <= i & i < i2) | ~(EX i. (i, v1) :
2613     sa..contents & 0 <= i & i < i2) & (i2, v1) : sa..contents & v1 = v2) | ~(EX
2614     i. (i, v1) : sa..contents & 0 <= i & i <= i2) & (EX i. (i, v1) :
2615     sa..contents & i2 < i & i < sa..csize) & v1 ~= v2)" */
2616     int r1a = sa.indexOf(v1);
2617     /*: assume "0 <= i2 & i2 < sa..csize" */
2618     Object r2a = sa.set(i2, v2);

```

```

2605     /*: assume "0 <= i2 & i2 < sb..csize" */
2606     Object r2b = sb.set(i2, v2);
2607     int r1b = sb.indexOf(v1);
2608
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 &
        sa..contents = sb..contents & sa..csize = sb..csize"
2615     modifies "sa..contents", "sb..contents"
2616     ensures "True" */
2617 {
2618     int r1a = sa.indexOf(v1);
2619     /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 = v2)
        | (r1a > i2 & v1 ~= v2)" */
2620     /*: assume "0 <= i2 & i2 < sa..csize" */
2621     Object r2a = sa.set(i2, v2);
2622
2623     Object r2b = sb.set(i2, v2);
2624     int r1b = sb.indexOf(v1);
2625
2626     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
2627 }
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 &
        sa..contents = sb..contents & sa..csize = sb..csize"
2632     modifies "sa..contents", "sb..contents"
2633     ensures "True" */
2634 {
2635     int r1a = sa.indexOf(v1);
2636     /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 =
        v2) | (r1a > i2 & v1 ~= v2))" */
2637     /*: assume "0 <= i2 & i2 < sa..csize" */
2638     Object r2a = sa.set(i2, v2);
2639
2640     /*: assume "0 <= i2 & i2 < sb..csize" */
2641     Object r2b = sb.set(i2, v2);
2642     int r1b = sb.indexOf(v1);
2643
2644     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
2645 }
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 &
        sa..contents = sb..contents & sa..csize = sb..csize"
2650     modifies "sa..contents", "sb..contents"
2651     ensures "True" */
2652 {
2653     int r1a = sa.indexOf(v1);
2654     /*: assume "0 <= i2 & i2 < sa..csize" */
2655     Object r2a = sa.set(i2, v2);
2656     /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 = v2)
        | (r1a > i2 & v1 ~= v2)" */
2657
2658     Object r2b = sb.set(i2, v2);
2659     int r1b = sb.indexOf(v1);
2660

```

```

2661     /*: assert "ria = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2662         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 =
2675         v2) | (r1a > i2 & v1 ~= v2))" */
2676
2677     /*: assume "0 <= i2 & i2 < sb..csize" */
2678     Object r2b = sb.set(i2, v2);
2679     int r1b = sb.indexOf(v1);
2680
2681     /*: assert "~(ria = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2682         sb..csize)" */
2683 }
2684
2685 static void indexOf_set_pre_s_75(ArrayList sa, ArrayList sb, Object v1, int i2,
    Object v2)
2686 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2687     sa..contents = sb..contents & sa..csize = sb..csize"
2688     modifies "sa..contents", "sb..contents"
2689     ensures "True" */
2690 {
2691     /*: assume "~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
2692         v2) | (EX i. (i, v1) : sa..contents & 0 <= i & i < i2) | (~(EX i. (i, v1) :
2693         sa..contents & 0 <= i & i < i2) & (i2, v1) : sa..contents & v1 = v2) | (~(EX
2694         i. (i, v1) : sa..contents & 0 <= i & i <= i2) & (EX i. (i, v1) :
2695         sa..contents & i2 < i & i < sa..csize) & v1 ~= v2)" */
2696     int r1a = sa.indexOf(v1);
2697     /*: assume "0 <= i2 & i2 < sa..csize" */
2698     sa.set(i2, v2);
2699
2700     sb.set(i2, v2);
2701     int r1b = sb.indexOf(v1);
2702
2703     /*: assert "ria = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
2704 }
2705
2706 static void indexOf_set_pre_c_75(ArrayList sa, ArrayList sb, Object v1, int i2,
    Object v2)
2707 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2708     sa..contents = sb..contents & sa..csize = sb..csize"
2709     modifies "sa..contents", "sb..contents"
2710     ensures "True" */
2711 {
2712     /*: assume "~(~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
2713         v2) | (EX i. (i, v1) : sa..contents & 0 <= i & i < i2) | (~(EX i. (i, v1) :
2714         sa..contents & 0 <= i & i < i2) & (i2, v1) : sa..contents & v1 = v2) | (~(EX
2715         i. (i, v1) : sa..contents & 0 <= i & i <= i2) & (EX i. (i, v1) :
2716         sa..contents & i2 < i & i < sa..csize) & v1 ~= v2))" */
2717     int r1a = sa.indexOf(v1);
2718     /*: assume "0 <= i2 & i2 < sa..csize" */
2719     sa.set(i2, v2);
2720
2721     /*: assume "0 <= i2 & i2 < sb..csize" */

```

```

2712     sb.set(i2, v2);
2713     int r1b = sb.indexOf(v1);
2714
2715     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
2716 }
2717
2718 static void indexOf_set_between_s_76(ArrayList sa, ArrayList sb, Object v1, int i2,
    Object v2)
2719 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2720     sa..contents = sb..contents & sa..csize = sb..csize"
2721     modifies "sa..contents", "sb..contents"
2722     ensures "True" */
2723 {
2724     int r1a = sa.indexOf(v1);
2725     /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 = v2)
        | (r1a > i2 & v1 ~= v2)" */
2726     /*: assume "0 <= i2 & i2 < sa..csize" */
2727     sa.set(i2, v2);
2728
2729     sb.set(i2, v2);
2730     int r1b = sb.indexOf(v1);
2731
2732     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
2733 }
2734
2735 static void indexOf_set_between_c_76(ArrayList sa, ArrayList sb, Object v1, int i2,
    Object v2)
2736 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2737     sa..contents = sb..contents & sa..csize = sb..csize"
2738     modifies "sa..contents", "sb..contents"
2739     ensures "True" */
2740 {
2741     int r1a = sa.indexOf(v1);
2742     /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 =
        v2) | (r1a > i2 & v1 ~= v2))" */
2743     /*: assume "0 <= i2 & i2 < sa..csize" */
2744     sa.set(i2, v2);
2745
2746     /*: assume "0 <= i2 & i2 < sb..csize" */
2747     sb.set(i2, v2);
2748     int r1b = sb.indexOf(v1);
2749
2750     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
2751 }
2752
2753 static void indexOf_set_post_s_77(ArrayList sa, ArrayList sb, Object v1, int i2,
    Object v2)
2754 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2755     sa..contents = sb..contents & sa..csize = sb..csize"
2756     modifies "sa..contents", "sb..contents"
2757     ensures "True" */
2758 {
2759     int r1a = sa.indexOf(v1);
2760     /*: assume "0 <= i2 & i2 < sa..csize" */
2761     sa.set(i2, v2);
2762     /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 = v2)
        | (r1a > i2 & v1 ~= v2)" */
2763
2764     sb.set(i2, v2);
2765     int r1b = sb.indexOf(v1);
2766
2767     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
2768 }

```

```

2769 static void indexOf_set_post_c_77(ArrayList sa, ArrayList sb, Object v1, int i2,
2770 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 =
2780 v2) | (r1a > i2 & v1 ~= v2))" */
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     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2830         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 =
2846         sb..csize)" */
2847 }
2848
2849 static void indexOf_size_post_s_80(ArrayList sa, ArrayList sb, Object v1)
2850 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2851     sa..contents = sb..contents & sa..csize = sb..csize"
2852     ensures "True" */
2853 {
2854     int r1a = sa.indexOf(v1);
2855     int r2a = sa.size();
2856     /*: assume "True" */
2857
2858     int r2b = sb.size();
2859     int r1b = sb.indexOf(v1);
2860
2861     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2862         sb..csize" */
2863 }
2864
2865 static void indexOf_size_post_c_80(ArrayList sa, ArrayList sb, Object v1)
2866 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2867     sa..contents = sb..contents & sa..csize = sb..csize"
2868     ensures "True" */
2869 {
2870     int r1a = sa.indexOf(v1);
2871     int r2a = sa.size();
2872     /*: assume "~(True)" */
2873
2874     int r2b = sb.size();
2875     int r1b = sb.indexOf(v1);
2876
2877     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2878         sb..csize)" */
2879 }
2880
2881 static void lastIndexOf_add_at_pre_s_81(ArrayList sa, ArrayList sb, Object v1, int
2882     i2, Object v2)
2883 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2884     sa..contents = sb..contents & sa..csize = sb..csize"
2885     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
2886         "sb..msize"
2887     ensures "True" */
2888 {
2889     /*: assume "~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
2890         v2) | ((EX i. (i, v1) : sa..contents & 0 <= i & i < i2) & ~(EX i. (i, v1) :
2891         sa..contents & i2 <= i & i < sa..csize) & v1 ~= v2)" */
2892     int r1a = sa.lastIndexOf(v1);

```



```

2886     /*: assume "0 <= i2 & i2 <= sa..csize" */
2887     sa.add_at(i2, v2);
2888
2889     sb.add_at(i2, v2);
2890     int r1b = sb.lastIndexOf(v1);
2891
2892     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
2893 }
2894
2895 static void lastIndexOf_add_at_pre_c_81(ArrayList sa, ArrayList sb, Object v1, int
2896     i2, Object v2)
2897 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2898     sa..contents = sb..contents & sa..csize = sb..csize"
2899     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
2900     "sb..msize"
2901     ensures "True" */
2902 {
2903     /*: assume "~((~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
2904     v2) | ((EX i. (i, v1) : sa..contents & 0 <= i & i < i2) & ~(EX i. (i, v1) :
2905     sa..contents & i2 <= i & i < sa..csize) & v1 ~= v2))" */
2906     int r1a = sa.lastIndexOf(v1);
2907     /*: assume "0 <= i2 & i2 <= sa..csize" */
2908     sa.add_at(i2, v2);
2909
2910     /*: assume "0 <= i2 & i2 <= sb..csize" */
2911     sb.add_at(i2, v2);
2912     int r1b = sb.lastIndexOf(v1);
2913
2914     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
2915     */
2916 }
2917
2918 static void lastIndexOf_add_at_between_s_82(ArrayList sa, ArrayList sb, Object v1,
2919     int i2, Object v2)
2920 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2921     sa..contents = sb..contents & sa..csize = sb..csize"
2922     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
2923     "sb..msize"
2924     ensures "True" */
2925 {
2926     int r1a = sa.lastIndexOf(v1);
2927     /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2)" */
2928     /*: assume "0 <= i2 & i2 <= sa..csize" */
2929     sa.add_at(i2, v2);
2930
2931     sb.add_at(i2, v2);
2932     int r1b = sb.lastIndexOf(v1);
2933
2934     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
2935 }
2936
2937 static void lastIndexOf_add_at_between_c_82(ArrayList sa, ArrayList sb, Object v1,
2938     int i2, Object v2)
2939 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2940     sa..contents = sb..contents & sa..csize = sb..csize"
2941     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
2942     "sb..msize"
2943     ensures "True" */
2944 {
2945     int r1a = sa.lastIndexOf(v1);
2946     /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2))" */
2947     /*: assume "0 <= i2 & i2 <= sa..csize" */
2948     sa.add_at(i2, v2);
2949
2950     /*: 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)

```

```

3000  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3001          sa..contents = sb..contents & sa..csize = sb..csize"
3002  ensures "True" */
3003  {
3004    /*: assume "~(True)" */
3005    int r1a = sa.lastIndexOf(v1);
3006    /*: assume "0 <= i2 & i2 < sa..csize" */
3007    Object r2a = sa.get(i2);
3008
3009    /*: assume "0 <= i2 & i2 < sb..csize" */
3010    Object r2b = sb.get(i2);
3011    int r1b = sb.lastIndexOf(v1);
3012
3013    /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3014          sb..csize)" */
3015  }
3016
3017  static void lastIndexOf_get_between_s_85(ArrayList sa, ArrayList sb, Object v1, int
3018  i2)
3019  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3020          sa..contents = sb..contents & sa..csize = sb..csize"
3021  ensures "True" */
3022  {
3023    int r1a = sa.lastIndexOf(v1);
3024    /*: assume "True" */
3025    /*: assume "0 <= i2 & i2 < sa..csize" */
3026    Object r2a = sa.get(i2);
3027
3028    Object r2b = sb.get(i2);
3029    int r1b = sb.lastIndexOf(v1);
3030
3031    /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3032          sb..csize" */
3033  }
3034
3035  static void lastIndexOf_get_between_c_85(ArrayList sa, ArrayList sb, Object v1, int
3036  i2)
3037  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3038          sa..contents = sb..contents & sa..csize = sb..csize"
3039  ensures "True" */
3040  {
3041    int r1a = sa.lastIndexOf(v1);
3042    /*: assume "~(True)" */
3043    /*: assume "0 <= i2 & i2 < sa..csize" */
3044    Object r2a = sa.get(i2);
3045
3046    /*: assume "0 <= i2 & i2 < sb..csize" */
3047    Object r2b = sb.get(i2);
3048    int r1b = sb.lastIndexOf(v1);
3049
3050    /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3051          sb..csize)" */
3052  }
3053
3054  static void lastIndexOf_get_post_s_86(ArrayList sa, ArrayList sb, Object v1, int i2)
3055  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3056          sa..contents = sb..contents & sa..csize = sb..csize"
3057  ensures "True" */
3058  {
3059    int r1a = sa.lastIndexOf(v1);
3060    /*: assume "0 <= i2 & i2 < sa..csize" */
3061    Object r2a = sa.get(i2);
3062    /*: assume "True" */
3063
3064    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 &
        sa..contents = sb..contents & sa..csize = sb..csize"
3067     ensures "True" */
3068 {
3069     int r1a = sa.lastIndexOf(v1);
3070     /*: assume "0 <= i2 & i2 < sa..csize" */
3071     Object r2a = sa.get(i2);
3072     /*: assume "~(True)" */
3073
3074     /*: assume "0 <= i2 & i2 < sb..csize" */
3075     Object r2b = sb.get(i2);
3076     int r1b = sb.lastIndexOf(v1);
3077
3078     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
3079 }
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 &
        sa..contents = sb..contents & sa..csize = sb..csize"
3084     ensures "True" */
3085 {
3086     /*: assume "True" */
3087     int r1a = sa.lastIndexOf(v1);
3088     int r2a = sa.indexOf(v2);
3089
3090     int r2b = sb.indexOf(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
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 &
        sa..contents = sb..contents & sa..csize = sb..csize"
3099     ensures "True" */
3100 {
3101     /*: assume "~(True)" */
3102     int r1a = sa.lastIndexOf(v1);
3103     int r2a = sa.indexOf(v2);
3104
3105     int r2b = sb.indexOf(v2);
3106     int r1b = sb.lastIndexOf(v1);
3107
3108     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
3109 }
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 &
        sa..contents = sb..contents & sa..csize = sb..csize"
3114     ensures "True" */
3115 {
3116     int r1a = sa.lastIndexOf(v1);
3117

```

```

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 =
3185         sb..csize" */
3186 }
3187
3188 static void lastIndexOf_lastIndexOf_pre_c_90(ArrayList sa, ArrayList sb, Object v1,
3189     Object v2)
3190 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3191     sa..contents = sb..contents & sa..csize = sb..csize"
3192     ensures "True" */
3193 {
3194     /*: assume "~(True)" */
3195     int r1a = sa.lastIndexOf(v1);
3196     int r2a = sa.lastIndexOf(v2);
3197
3198     int r2b = sb.lastIndexOf(v2);
3199     int r1b = sb.lastIndexOf(v1);
3200
3201     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3202         sb..csize)" */
3203 }
3204
3205 static void lastIndexOf_lastIndexOf_between_s_91(ArrayList sa, ArrayList sb, Object
3206     v1, Object v2)
3207 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3208     sa..contents = sb..contents & sa..csize = sb..csize"
3209     ensures "True" */
3210 {
3211     int r1a = sa.lastIndexOf(v1);
3212     /*: assume "True" */
3213     int r2a = sa.lastIndexOf(v2);
3214
3215     int r2b = sb.lastIndexOf(v2);
3216     int r1b = sb.lastIndexOf(v1);
3217
3218     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3219         sb..csize" */
3220 }
3221
3222 static void lastIndexOf_lastIndexOf_between_c_91(ArrayList sa, ArrayList sb, Object
3223     v1, Object v2)
3224 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3225     sa..contents = sb..contents & sa..csize = sb..csize"
3226     ensures "True" */
3227 {
3228     int r1a = sa.lastIndexOf(v1);
3229     /*: assume "~(True)" */
3230     int r2a = sa.lastIndexOf(v2);
3231
3232     int r2b = sb.lastIndexOf(v2);
3233     int r1b = sb.lastIndexOf(v1);
3234
3235     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3236         sb..csize)" */
3237 }

```

```

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 =
3295         sb..csize)" */
3296 }
3297
3298 static void lastIndexOf_remove_at_between_s_94(ArrayList sa, ArrayList sb, Object
3299     v1, int i2)
3300 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3301     sa..contents = sb..contents & sa..csize = sb..csize"
3302 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3303 ensures "True" */
3304 {
3305     int r1a = sa.lastIndexOf(v1);
3306     /*: assume "r1a < 0 | (0 <= r1a & r1a < i2)" */
3307     /*: assume "0 <= i2 & i2 < sa..csize" */
3308     Object r2a = sa.remove_at(i2);
3309
3310     Object r2b = sb.remove_at(i2);
3311     int r1b = sb.lastIndexOf(v1);
3312
3313     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3314         sb..csize" */
3315 }
3316
3317 static void lastIndexOf_remove_at_between_c_94(ArrayList sa, ArrayList sb, Object
3318     v1, int i2)
3319 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3320     sa..contents = sb..contents & sa..csize = sb..csize"
3321 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3322 ensures "True" */
3323 {
3324     int r1a = sa.lastIndexOf(v1);
3325     /*: assume "~(r1a < 0 | (0 <= r1a & r1a < i2))" */
3326     /*: assume "0 <= i2 & i2 < sa..csize" */
3327     Object r2a = sa.remove_at(i2);
3328
3329     /*: assume "0 <= i2 & i2 < sb..csize" */
3330     Object r2b = sb.remove_at(i2);
3331     int r1b = sb.lastIndexOf(v1);
3332
3333     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3334         sb..csize)" */
3335 }
3336
3337 static void lastIndexOf_remove_at_post_s_95(ArrayList sa, ArrayList sb, Object v1,
3338     int i2)
3339 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3340     sa..contents = sb..contents & sa..csize = sb..csize"
3341 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3342 ensures "True" */
3343 {
3344     int r1a = sa.lastIndexOf(v1);
3345     /*: assume "0 <= i2 & i2 < sa..csize" */
3346     Object r2a = sa.remove_at(i2);
3347     /*: assume "r1a < 0 | (0 <= r1a & r1a < i2)" */
3348
3349     Object r2b = sb.remove_at(i2);
3350     int r1b = sb.lastIndexOf(v1);

```



```

3345     /*: assert "ria = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3346         sb..csize" */
3347 }
3348
3349 static void lastIndexOf_remove_at_post_c_95(ArrayList sa, ArrayList sb, Object v1,
3350     int i2)
3351 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3352     sa..contents = sb..contents & sa..csize = sb..csize"
3353     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3354     ensures "True" */
3355 {
3356     int r1a = sa.lastIndexOf(v1);
3357     /*: assume "0 <= i2 & i2 < sa..csize" */
3358     Object r2a = sa.remove_at(i2);
3359     /*: assume "~(ria < 0 | (0 <= r1a & r1a < i2))" */
3360
3361     /*: assume "0 <= i2 & i2 < sb..csize" */
3362     Object r2b = sb.remove_at(i2);
3363     int r1b = sb.lastIndexOf(v1);
3364
3365     /*: assert "~(ria = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3366         sb..csize)" */
3367 }
3368
3369 static void lastIndexOf_remove_at_pre_s_96(ArrayList sa, ArrayList sb, Object v1,
3370     int i2)
3371 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3372     sa..contents = sb..contents & sa..csize = sb..csize"
3373     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3374     ensures "True" */
3375 {
3376     /*: assume "~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) | ((EX i.
3377         (i, v1) : sa..contents & 0 <= i & i < i2) & ~(EX i. (i, v1) : sa..contents &
3378         i2 <= i & i < sa..csize))" */
3379     int r1a = sa.lastIndexOf(v1);
3380     /*: assume "0 <= i2 & i2 < sa..csize" */
3381     sa.remove_at(i2);
3382
3383     sb.remove_at(i2);
3384     int r1b = sb.lastIndexOf(v1);
3385
3386     /*: assert "ria = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
3387 }
3388
3389 static void lastIndexOf_remove_at_pre_c_96(ArrayList sa, ArrayList sb, Object v1,
3390     int i2)
3391 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3392     sa..contents = sb..contents & sa..csize = sb..csize"
3393     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3394     ensures "True" */
3395 {
3396     /*: assume "~(~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) | ((EX
3397         i. (i, v1) : sa..contents & 0 <= i & i < i2) & ~(EX i. (i, v1) :
3398         sa..contents & i2 <= i & i < sa..csize)))" */
3399     int r1a = sa.lastIndexOf(v1);
3400     /*: assume "0 <= i2 & i2 < sa..csize" */
3401     sa.remove_at(i2);
3402
3403     /*: assume "0 <= i2 & i2 < sb..csize" */
3404     sb.remove_at(i2);
3405     int r1b = sb.lastIndexOf(v1);
3406
3407     /*: assert "~(ria = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
3408         */

```

```

3400 }
3401
3402 static void lastIndexOf_remove_at_between_s_97(ArrayList sa, ArrayList sb, Object
3403     v1, int i2)
3404 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3405     sa..contents = sb..contents & sa..csize = sb..csize"
3406     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3407     ensures "True" */
3408 {
3409     int r1a = sa.lastIndexOf(v1);
3410     /*: assume "r1a < 0 | (0 <= r1a & r1a < i2)" */
3411     /*: assume "0 <= i2 & i2 < sa..csize" */
3412     sa.remove_at(i2);
3413
3414     sb.remove_at(i2);
3415     int r1b = sb.lastIndexOf(v1);
3416
3417     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
3418 }
3419
3420 static void lastIndexOf_remove_at_between_c_97(ArrayList sa, ArrayList sb, Object
3421     v1, int i2)
3422 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3423     sa..contents = sb..contents & sa..csize = sb..csize"
3424     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3425     ensures "True" */
3426 {
3427     int r1a = sa.lastIndexOf(v1);
3428     /*: assume "~(r1a < 0 | (0 <= r1a & r1a < i2))" */
3429     /*: assume "0 <= i2 & i2 < sa..csize" */
3430     sa.remove_at(i2);
3431
3432     /*: assume "0 <= i2 & i2 < sb..csize" */
3433     sb.remove_at(i2);
3434     int r1b = sb.lastIndexOf(v1);
3435
3436     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
3437     */
3438 }
3439
3440 static void lastIndexOf_remove_at_post_s_98(ArrayList sa, ArrayList sb, Object v1,
3441     int i2)
3442 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3443     sa..contents = sb..contents & sa..csize = sb..csize"
3444     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3445     ensures "True" */
3446 {
3447     int r1a = sa.lastIndexOf(v1);
3448     /*: assume "0 <= i2 & i2 < sa..csize" */
3449     sa.remove_at(i2);
3450     /*: assume "r1a < 0 | (0 <= r1a & r1a < i2)" */
3451
3452     sb.remove_at(i2);
3453     int r1b = sb.lastIndexOf(v1);
3454
3455     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
3456 }
3457
3458 static void lastIndexOf_remove_at_post_c_98(ArrayList sa, ArrayList sb, Object v1,
3459     int i2)
3460 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3461     sa..contents = sb..contents & sa..csize = sb..csize"
3462     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3463     ensures "True" */
3464 {

```

```

3460     int r1a = sa.lastIndexOf(v1);
3461     /*: assume "0 <= i2 & i2 < sa..csize" */
3462     sa.remove_at(i2);
3463     /*: assume "~(r1a < 0 | (0 <= r1a & r1a < i2))" */
3464
3465     /*: assume "0 <= i2 & i2 < sb..csize" */
3466     sb.remove_at(i2);
3467     int r1b = sb.lastIndexOf(v1);
3468
3469     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
3470 }
3471
3472 static void lastIndexOf_set_pre_s_99(ArrayList sa, ArrayList sb, Object v1, int i2,
    Object v2)
3473 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3474     sa..contents = sb..contents & sa..csize = sb..csize"
3475     modifies "sa..contents", "sb..contents"
3476     ensures "True" */
3477 {
3478     /*: 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))" */
3479     int r1a = sa.lastIndexOf(v1);
3480     /*: assume "0 <= i2 & i2 < sa..csize" */
3481     Object r2a = sa.set(i2, v2);
3482
3483     Object r2b = sb.set(i2, v2);
3484     int r1b = sb.lastIndexOf(v1);
3485
3486     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
3487 }
3488
3489 static void lastIndexOf_set_pre_c_99(ArrayList sa, ArrayList sb, Object v1, int i2,
    Object v2)
3490 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3491     sa..contents = sb..contents & sa..csize = sb..csize"
3492     modifies "sa..contents", "sb..contents"
3493     ensures "True" */
3494 {
3495     /*: 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)))" */
3496     int r1a = sa.lastIndexOf(v1);
3497     /*: assume "0 <= i2 & i2 < sa..csize" */
3498     Object r2a = sa.set(i2, v2);
3499
3500     /*: assume "0 <= i2 & i2 < sb..csize" */
3501     Object r2b = sb.set(i2, v2);
3502     int r1b = sb.lastIndexOf(v1);
3503
3504     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
3505 }
3506
3507 static void lastIndexOf_set_between_s_100(ArrayList sa, ArrayList sb, Object v1,
    int i2, Object v2)
3508 /*: 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 {

```

```

3565     int r1a = sa.lastIndexOf(v1);
3566     /*: assume "0 <= i2 & i2 < sa..csize" */
3567     Object r2a = sa.set(i2, v2);
3568     /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2) | (r1a =
        i2 & v1 = v2) | r1a > i2)" */
3569
3570     /*: assume "0 <= i2 & i2 < sb..csize" */
3571     Object r2b = sb.set(i2, v2);
3572     int r1b = sb.lastIndexOf(v1);
3573
3574     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
3575 }
3576
3577 static void lastIndexOf_set_pre_s_102(ArrayList sa, ArrayList sb, Object v1, int
    i2, Object v2)
3578 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3579     sa..contents = sb..contents & sa..csize = sb..csize"
3580 modifies "sa..contents", "sb..contents"
3581 ensures "True" */
3582 {
3583     /*: 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)" */
3584     int r1a = sa.lastIndexOf(v1);
3585     /*: assume "0 <= i2 & i2 < sa..csize" */
3586     sa.set(i2, v2);
3587
3588     sb.set(i2, v2);
3589     int r1b = sb.lastIndexOf(v1);
3590
3591     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
3592 }
3593
3594 static void lastIndexOf_set_pre_c_102(ArrayList sa, ArrayList sb, Object v1, int
    i2, Object v2)
3595 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3596     sa..contents = sb..contents & sa..csize = sb..csize"
3597 modifies "sa..contents", "sb..contents"
3598 ensures "True" */
3599 {
3600     /*: 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))" */
3601     int r1a = sa.lastIndexOf(v1);
3602     /*: assume "0 <= i2 & i2 < sa..csize" */
3603     sa.set(i2, v2);
3604
3605     /*: assume "0 <= i2 & i2 < sb..csize" */
3606     sb.set(i2, v2);
3607     int r1b = sb.lastIndexOf(v1);
3608
3609     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
3610 }
3611
3612 static void lastIndexOf_set_between_s_103(ArrayList sa, ArrayList sb, Object v1,
    int i2, Object v2)
3613 /*: 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
3620         & v1 = v2) | r1a > i2" */
3621     /*: assume "0 <= i2 & i2 < sa..csize" */
3622     sa.set(i2, v2);
3623
3624     sb.set(i2, v2);
3625     int r1b = sb.lastIndexOf(v1);
3626
3627     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
3628 }
3629
3630 static void lastIndexOf_set_between_c_103(ArrayList sa, ArrayList sb, Object v1,
3631     int i2, Object v2)
3632 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3633     sa..contents = sb..contents & sa..csize = sb..csize"
3634     modifies "sa..contents", "sb..contents"
3635     ensures "True" */
3636 {
3637     int r1a = sa.lastIndexOf(v1);
3638     /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2) | (r1a =
3639         i2 & v1 = v2) | r1a > i2)" */
3640     /*: assume "0 <= i2 & i2 < sa..csize" */
3641     sa.set(i2, v2);
3642
3643     /*: assume "0 <= i2 & i2 < sb..csize" */
3644     sb.set(i2, v2);
3645     int r1b = sb.lastIndexOf(v1);
3646
3647     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
3648         */
3649 }
3650
3651 static void lastIndexOf_set_post_s_104(ArrayList sa, ArrayList sb, Object v1, int
3652     i2, Object v2)
3653 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3654     sa..contents = sb..contents & sa..csize = sb..csize"
3655     modifies "sa..contents", "sb..contents"
3656     ensures "True" */
3657 {
3658     int r1a = sa.lastIndexOf(v1);
3659     /*: assume "0 <= i2 & i2 < sa..csize" */
3660     sa.set(i2, v2);
3661     /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2) | (r1a = i2
3662         & v1 = v2) | r1a > i2" */
3663
3664     sb.set(i2, v2);
3665     int r1b = sb.lastIndexOf(v1);
3666
3667     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
3668 }
3669
3670 static void lastIndexOf_set_post_c_104(ArrayList sa, ArrayList sb, Object v1, int
3671     i2, Object v2)
3672 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3673     sa..contents = sb..contents & sa..csize = sb..csize"
3674     modifies "sa..contents", "sb..contents"
3675     ensures "True" */
3676 {
3677     int r1a = sa.lastIndexOf(v1);
3678     /*: assume "0 <= i2 & i2 < sa..csize" */

```

```

3672     sa.set(i2, v2);
3673     /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2) | (r1a =
        i2 & v1 = v2) | r1a > i2)" */
3674
3675     /*: assume "0 <= i2 & i2 < sb..csize" */
3676     sb.set(i2, v2);
3677     int r1b = sb.lastIndexOf(v1);
3678
3679     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
3680 }
3681
3682 static void lastIndexOf_size_pre_s_105(ArrayList sa, ArrayList sb, Object v1)
3683 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
3684 {
3685     /*: assume "True" */
3686     int r1a = sa.lastIndexOf(v1);
3687     int r2a = sa.size();
3688
3689     int r2b = sb.size();
3690     int r1b = sb.lastIndexOf(v1);
3691
3692     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
3693 }
3694
3695 static void lastIndexOf_size_pre_c_105(ArrayList sa, ArrayList sb, Object v1)
3696 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
3697 {
3698     /*: assume "~(True)" */
3699     int r1a = sa.lastIndexOf(v1);
3700     int r2a = sa.size();
3701
3702     int r2b = sb.size();
3703     int r1b = sb.lastIndexOf(v1);
3704
3705     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
3706 }
3707
3708 static void lastIndexOf_size_between_s_106(ArrayList sa, ArrayList sb, Object v1)
3709 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
3710 {
3711     int r1a = sa.lastIndexOf(v1);
3712     /*: assume "True" */
3713     int r2a = sa.size();
3714
3715     int r2b = sb.size();
3716     int r1b = sb.lastIndexOf(v1);
3717
3718     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
3719 }
3720
3721 static void lastIndexOf_size_between_c_106(ArrayList sa, ArrayList sb, Object v1)
3722 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
        ensures "True" */
3723 {
3724
3725
3726
3727
3728
3729
3730
3731

```

```

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 static void remove_at_add_at_pre_c_108(ArrayList sa, ArrayList sb, int i1, int i2,
3790     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",
3794     "sb..msize"
3795     ensures "True" */
3796 {
3797     /*: assume "~((i1 < i2 & (i2, v2) : sa..contents & 0 <= i2 & i2 < sa..csize) |
3798     (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize) | (i1 > i2 &
3799     (ALL v. ((i1 - 1, v) : sa..contents) = ((i1, v) : sa..contents)) & 0 <= i1 -
3800     1 & i1 - 1 < sa..csize & 0 <= i1 & i1 < sa..csize))" */
3801     /*: assume "0 <= i1 & i1 < sa..csize" */
3802     Object r1a = sa.remove_at(i1);
3803     /*: assume "0 <= i2 & i2 <= sa..csize" */
3804     sa.add_at(i2, v2);
3805
3806     /*: assume "0 <= i2 & i2 <= sb..csize" */
3807     sb.add_at(i2, v2);
3808     /*: assume "0 <= i1 & i1 < sb..csize" */
3809     Object r1b = sb.remove_at(i1);
3810
3811     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
3812     */
3813 }
3814
3815 static void remove_at_add_at_between_s_109(ArrayList sa, ArrayList sb, int i1, int
3816     i2, Object v2)
3817 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3818     sa..contents = sb..contents & sa..csize = sb..csize"
3819     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
3820     "sb..msize"
3821     ensures "True" */
3822 {
3823     /*: assume "0 <= i1 & i1 < sa..csize" */
3824     Object r1a = sa.remove_at(i1);
3825     /*: assume "(i1 < i2 & (i2 - 1, v2) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
3826     sa..csize) | (i1 = i2 & r1a = v2) | (i1 > i2 & (i1 - 1, r1a) : sa..contents
3827     & 0 <= i1 - 1 & i1 - 1 < sa..csize)" */
3828     /*: assume "0 <= i2 & i2 <= sa..csize" */
3829     sa.add_at(i2, v2);
3830
3831     sb.add_at(i2, v2);
3832     Object r1b = sb.remove_at(i1);
3833
3834     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
3835 }
3836
3837 static void remove_at_add_at_between_c_109(ArrayList sa, ArrayList sb, int i1, int
3838     i2, Object v2)
3839 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3840     sa..contents = sb..contents & sa..csize = sb..csize"
3841     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
3842     "sb..msize"
3843     ensures "True" */
3844 {
3845     /*: assume "0 <= i1 & i1 < sa..csize" */
3846     Object r1a = sa.remove_at(i1);
3847     /*: assume "~((i1 < i2 & (i2 - 1, v2) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
3848     sa..csize) | (i1 = i2 & r1a = v2) | (i1 > i2 & (i1 - 1, r1a) : sa..contents
3849     & 0 <= i1 - 1 & i1 - 1 < sa..csize))" */
3850     /*: assume "0 <= i2 & i2 <= sa..csize" */
3851     sa.add_at(i2, v2);

```

```

3840     /*: assume "0 <= i2 & i2 <= sb..csize" */
3841     sb.add_at(i2, v2);
3842     /*: assume "0 <= i1 & i1 < sb..csize" */
3843     Object r1b = sb.remove_at(i1);
3844
3845     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
3846 }
3847
3848 static void remove_at_add_at_post_s_110(ArrayList sa, ArrayList sb, int i1, int i2,
3849     Object v2)
3850 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3851     sa..contents = sb..contents & sa..csize = sb..csize"
3852     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
3853     "sb..msize"
3854     ensures "True" */
3855 {
3856     /*: assume "0 <= i1 & i1 < sa..csize" */
3857     Object r1a = sa.remove_at(i1);
3858     /*: assume "0 <= i2 & i2 <= sa..csize" */
3859     sa.add_at(i2, v2);
3860     /*: assume "(i1 < i2 & (i2 - 1, v2) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
3861     sa..csize) | (i1 = i2 & r1a = v2) | (i1 > i2 & (i1, r1a) : sa..contents & 0
3862     <= i1 & i1 < sa..csize)" */
3863
3864     sb.add_at(i2, v2);
3865     Object r1b = sb.remove_at(i1);
3866
3867     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
3868 }
3869
3870 static void remove_at_add_at_post_c_110(ArrayList sa, ArrayList sb, int i1, int i2,
3871     Object v2)
3872 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3873     sa..contents = sb..contents & sa..csize = sb..csize"
3874     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
3875     "sb..msize"
3876     ensures "True" */
3877 {
3878     /*: assume "0 <= i1 & i1 < sa..csize" */
3879     Object r1a = sa.remove_at(i1);
3880     /*: assume "0 <= i2 & i2 <= sa..csize" */
3881     sa.add_at(i2, v2);
3882     /*: assume "~((i1 < i2 & (i2 - 1, v2) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
3883     sa..csize) | (i1 = i2 & r1a = v2) | (i1 > i2 & (i1, r1a) : sa..contents & 0
3884     <= i1 & i1 < sa..csize))" */
3885
3886     /*: assume "0 <= i2 & i2 <= sb..csize" */
3887     sb.add_at(i2, v2);
3888     /*: assume "0 <= i1 & i1 < sb..csize" */
3889     Object r1b = sb.remove_at(i1);
3890
3891     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
3892 }
3893
3894 static void remove_at_get_pre_s_111(ArrayList sa, ArrayList sb, int i1, int i2)
3895 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3896     sa..contents = sb..contents & sa..csize = sb..csize"
3897     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3898     ensures "True" */
3899 {
3900     /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
3901     sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
3902     sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :

```

```

3893     sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
3894     sa..csize) | i1 > i2" */
3895     /*: assume "0 <= i1 & i1 < sa..csize" */
3896     Object r1a = sa.remove_at(i1);
3897     /*: assume "0 <= i2 & i2 < sa..csize" */
3898     Object r2a = sa.get(i2);
3899     Object r2b = sb.get(i2);
3900     Object r1b = sb.remove_at(i1);
3901     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3902     sb..csize" */
3903 }
3904
3905 static void remove_at_get_pre_c_111(ArrayList sa, ArrayList sb, int i1, int i2)
3906 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3907     sa..contents = sb..contents & sa..csize = sb..csize"
3908     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3909     ensures "True" */
3910 {
3911     /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
3912     sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
3913     sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :
3914     sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
3915     sa..csize) | i1 > i2)" */
3916     /*: assume "0 <= i1 & i1 < sa..csize" */
3917     Object r1a = sa.remove_at(i1);
3918     /*: assume "0 <= i2 & i2 < sa..csize" */
3919     Object r2a = sa.get(i2);
3920     /*: assume "0 <= i2 & i2 < sb..csize" */
3921     Object r2b = sb.get(i2);
3922     /*: assume "0 <= i1 & i1 < sb..csize" */
3923     Object r1b = sb.remove_at(i1);
3924     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3925     sb..csize)" */
3926 }
3927
3928 static void remove_at_get_between_s_112(ArrayList sa, ArrayList sb, int i1, int i2)
3929 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3930     sa..contents = sb..contents & sa..csize = sb..csize"
3931     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3932     ensures "True" */
3933 {
3934     /*: assume "0 <= i1 & i1 < sa..csize" */
3935     Object r1a = sa.remove_at(i1);
3936     /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
3937     sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
3938     sa..csize) | (i1 = i2 & (i2, r1a) : sa..contents) | i1 > i2" */
3939     /*: assume "0 <= i2 & i2 < sa..csize" */
3940     Object r2a = sa.get(i2);
3941     Object r2b = sb.get(i2);
3942     Object r1b = sb.remove_at(i1);
3943     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3944     sb..csize" */
3945 }
3946
3947 static void remove_at_get_between_c_112(ArrayList sa, ArrayList sb, int i1, int i2)
3948 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3949     sa..contents = sb..contents & sa..csize = sb..csize"
3950     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3951     ensures "True" */

```

```

3947 {
3948     /*: assume "0 <= i1 & i1 < sa..csize" */
3949     Object r1a = sa.remove_at(i1);
3950     /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
3951         sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
3952         sa..csize) | (i1 = i2 & (i2, r1a) : sa..contents) | i1 > i2)" */
3953     /*: assume "0 <= i2 & i2 < sa..csize" */
3954     Object r2a = sa.get(i2);
3955
3956     /*: assume "0 <= i2 & i2 < sb..csize" */
3957     Object r2b = sb.get(i2);
3958     /*: assume "0 <= i1 & i1 < sb..csize" */
3959     Object r1b = sb.remove_at(i1);
3960
3961     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3962         sb..csize)" */
3963 }
3964
3965 static void remove_at_get_post_s_113(ArrayList sa, ArrayList sb, int i1, int i2)
3966 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3967     sa..contents = sb..contents & sa..csize = sb..csize"
3968     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3969     ensures "True" */
3970 {
3971     /*: assume "0 <= i1 & i1 < sa..csize" */
3972     Object r1a = sa.remove_at(i1);
3973     /*: assume "0 <= i2 & i2 < sa..csize" */
3974     Object r2a = sa.get(i2);
3975     /*: assume "(i1 < i2 & (i2 - 1, r2a) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
3976         sa..csize) | (i1 = i2 & r1a = r2a) | i1 > i2" */
3977
3978     Object r2b = sb.get(i2);
3979     Object r1b = sb.remove_at(i1);
3980
3981     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3982         sb..csize" */
3983 }
3984
3985 static void remove_at_get_post_c_113(ArrayList sa, ArrayList sb, int i1, int i2)
3986 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3987     sa..contents = sb..contents & sa..csize = sb..csize"
3988     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3989     ensures "True" */
3990 {
3991     /*: assume "0 <= i1 & i1 < sa..csize" */
3992     Object r1a = sa.remove_at(i1);
3993     /*: assume "0 <= i2 & i2 < sa..csize" */
3994     Object r2a = sa.get(i2);
3995     /*: assume "~((i1 < i2 & (i2 - 1, r2a) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
3996         sa..csize) | (i1 = i2 & r1a = r2a) | i1 > i2)" */
3997
3998     /*: assume "0 <= i2 & i2 < sb..csize" */
3999     Object r2b = sb.get(i2);
4000     /*: assume "0 <= i1 & i1 < sb..csize" */
4001     Object r1b = sb.remove_at(i1);
4002
4003     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
4004         sb..csize)" */
4005 }
4006
4007 static void remove_at_indexOf_pre_s_114(ArrayList sa, ArrayList sb, int i1, Object
4008     v2)
4009 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4010     sa..contents = sb..contents & sa..csize = sb..csize"
4011     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

```

```

4052 static void remove_at_indexOf_between_c_115(ArrayList sa, ArrayList sb, int i1,
4053 Object v2)
4054 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4055 sa..contents = sb..contents & sa..csize = sb..csize"
4056 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4057 ensures "True" */
4058 {
4059 /*: assume "0 <= i1 & i1 < sa..csize" */
4060 Object r1a = sa.remove_at(i1);
4061 /*: assume "~((EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & r1a
4062 ~= v2) | (EX i. (i, v2) : sa..contents & 0 <= i & i < i1) | ~(EX i. (i, v2)
4063 : sa..contents & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 <
4064 sa..csize & r1a = v2 & i1 < sa..csize))" */
4065 int r2a = sa.indexOf(v2);
4066
4067 int r2b = sb.indexOf(v2);
4068 /*: assume "0 <= i1 & i1 < sb..csize" */
4069 Object r1b = sb.remove_at(i1);
4070
4071 /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
4072 sb..csize)" */
4073 }
4074
4075 static void remove_at_indexOf_post_s_116(ArrayList sa, ArrayList sb, int i1, Object
4076 v2)
4077 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4078 sa..contents = sb..contents & sa..csize = sb..csize"
4079 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4080 ensures "True" */
4081 {
4082 /*: assume "0 <= i1 & i1 < sa..csize" */
4083 Object r1a = sa.remove_at(i1);
4084 int r2a = sa.indexOf(v2);
4085 /*: assume "(r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1) | (r2a = i1 & r1a =
4086 v2 & i1 < sa..csize)" */
4087
4088 int r2b = sb.indexOf(v2);
4089 Object r1b = sb.remove_at(i1);
4090
4091 /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
4092 sb..csize" */
4093 }
4094
4095 static void remove_at_indexOf_post_c_116(ArrayList sa, ArrayList sb, int i1, Object
4096 v2)
4097 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4098 sa..contents = sb..contents & sa..csize = sb..csize"
4099 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4100 ensures "True" */
4101 {
4102 /*: assume "0 <= i1 & i1 < sa..csize" */
4103 Object r1a = sa.remove_at(i1);
4104 int r2a = sa.indexOf(v2);
4105 /*: assume "~((r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1) | (r2a = i1 & r1a =
4106 v2 & i1 < sa..csize))" */
4107
4108 int r2b = sb.indexOf(v2);
4109 /*: assume "0 <= i1 & i1 < sb..csize" */
4110 Object r1b = sb.remove_at(i1);
4111
4112 /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
4113 sb..csize)" */
4114 }

```

```

4105 static void remove_at_lastIndexOf_pre_s_117(ArrayList sa, ArrayList sb, int i1,
4106         Object v2)
4107 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4108         sa..contents = sb..contents & sa..csize = sb..csize"
4109         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4110         ensures "True" */
4111 {
4112     /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | ((EX i.
4113         (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) : sa..contents &
4114         i1 <= i & i < sa..csize))" */
4115     /*: assume "0 <= i1 & i1 < sa..csize" */
4116     Object r1a = sa.remove_at(i1);
4117     int r2a = sa.lastIndexOf(v2);
4118
4119     int r2b = sb.lastIndexOf(v2);
4120     Object r1b = sb.remove_at(i1);
4121
4122     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
4123         sb..csize" */
4124 }
4125
4126 static void remove_at_lastIndexOf_pre_c_117(ArrayList sa, ArrayList sb, int i1,
4127         Object v2)
4128 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4129         sa..contents = sb..contents & sa..csize = sb..csize"
4130         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4131         ensures "True" */
4132 {
4133     /*: assume "~(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | ((EX
4134         i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
4135         sa..contents & i1 <= i & i < sa..csize))))" */
4136     /*: assume "0 <= i1 & i1 < sa..csize" */
4137     Object r1a = sa.remove_at(i1);
4138     int r2a = sa.lastIndexOf(v2);
4139
4140     int r2b = sb.lastIndexOf(v2);
4141     /*: assume "0 <= i1 & i1 < sb..csize" */
4142     Object r1b = sb.remove_at(i1);
4143
4144     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
4145         sb..csize)" */
4146 }
4147
4148 static void remove_at_lastIndexOf_between_s_118(ArrayList sa, ArrayList sb, int i1,
4149         Object v2)
4150 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4151         sa..contents = sb..contents & sa..csize = sb..csize"
4152         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4153         ensures "True" */
4154 {
4155     /*: assume "0 <= i1 & i1 < sa..csize" */
4156     Object r1a = sa.remove_at(i1);
4157     /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & r1a ~=
4158         v2 | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
4159         sa..contents & i1 <= i & i < sa..csize) & r1a ~= v2)" */
4160     int r2a = sa.lastIndexOf(v2);
4161
4162     int r2b = sb.lastIndexOf(v2);
4163     Object r1b = sb.remove_at(i1);
4164
4165     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
4166         sb..csize" */
4167 }

```

```

4157 static void remove_at_lastIndexOf_between_c_118(ArrayList sa, ArrayList sb, int i1,
4158         Object v2)
4159 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4160         sa..contents = sb..contents & sa..csize = sb..csize"
4161         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4162         ensures "True" */
4163 {
4164     /*: assume "0 <= i1 & i1 < sa..csize" */
4165     Object r1a = sa.remove_at(i1);
4166     /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & r1a
4167         ~= v2) | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2)
4168         : sa..contents & i1 <= i & i < sa..csize) & r1a ~= v2))" */
4169     int r2a = sa.lastIndexOf(v2);
4170
4171     int r2b = sb.lastIndexOf(v2);
4172     /*: assume "0 <= i1 & i1 < sb..csize" */
4173     Object r1b = sb.remove_at(i1);
4174
4175     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
4176         sb..csize)" */
4177 }
4178
4179 static void remove_at_lastIndexOf_post_s_119(ArrayList sa, ArrayList sb, int i1,
4180         Object v2)
4181 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4182         sa..contents = sb..contents & sa..csize = sb..csize"
4183         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4184         ensures "True" */
4185 {
4186     /*: assume "0 <= i1 & i1 < sa..csize" */
4187     Object r1a = sa.remove_at(i1);
4188     int r2a = sa.lastIndexOf(v2);
4189     /*: assume "(r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1 & r1a ~= v2)" */
4190
4191     int r2b = sb.lastIndexOf(v2);
4192     Object r1b = sb.remove_at(i1);
4193
4194     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
4195         sb..csize" */
4196 }
4197
4198 static void remove_at_lastIndexOf_post_c_119(ArrayList sa, ArrayList sb, int i1,
4199         Object v2)
4200 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4201         sa..contents = sb..contents & sa..csize = sb..csize"
4202         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4203         ensures "True" */
4204 {
4205     /*: assume "0 <= i1 & i1 < sa..csize" */
4206     Object r1a = sa.remove_at(i1);
4207     int r2a = sa.lastIndexOf(v2);
4208     /*: assume "~((r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1 & r1a ~= v2))" */
4209
4210     int r2b = sb.lastIndexOf(v2);
4211     /*: assume "0 <= i1 & i1 < sb..csize" */
4212     Object r1b = sb.remove_at(i1);
4213
4214     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
4215         sb..csize)" */
4216 }
4217
4218 static void remove_at_remove_at_pre_s_120(ArrayList sa, ArrayList sb, int i1, int
4219         i2)
4220 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4221         sa..contents = sb..contents & sa..csize = sb..csize"

```



```

4213     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4214     ensures "True" */
4215     {
4216     /*: 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)" */
4217     /*: assume "0 <= i1 & i1 < sa..csize" */
4218     Object r1a = sa.remove_at(i1);
4219     /*: assume "0 <= i2 & i2 < sa..csize" */
4220     Object r2a = sa.remove_at(i2);
4221
4222     Object r2b = sb.remove_at(i2);
4223     Object r1b = sb.remove_at(i1);
4224
4225     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
sb..csize" */
4226     }
4227
4228     static void remove_at_remove_at_pre_c_120(ArrayList sa, ArrayList sb, int i1, int
i2)
4229     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
sa..contents = sb..contents & sa..csize = sb..csize"
4230     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4231     ensures "True" */
4232     {
4233     /*: 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))" */
4235     /*: assume "0 <= i1 & i1 < sa..csize" */
4236     Object r1a = sa.remove_at(i1);
4237     /*: assume "0 <= i2 & i2 < sa..csize" */
4238     Object r2a = sa.remove_at(i2);
4239
4240     /*: assume "0 <= i2 & i2 < sb..csize" */
4241     Object r2b = sb.remove_at(i2);
4242     /*: assume "0 <= i1 & i1 < sb..csize" */
4243     Object r1b = sb.remove_at(i1);
4244
4245     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
sb..csize)" */
4246     }
4247
4248     static void remove_at_remove_at_between_s_121(ArrayList sa, ArrayList sb, int i1,
int i2)
4249     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
sa..contents = sb..contents & sa..csize = sb..csize"
4250     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4251     ensures "True" */
4252     {
4253     /*: assume "0 <= i1 & i1 < sa..csize" */
4254     Object r1a = sa.remove_at(i1);
4255     /*: 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)" */
4256     /*: 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" */

```

```

4311     Object r1a = sa.remove_at(i1);
4312     /*: assume "0 <= i2 & i2 < sa..csize" */
4313     Object r2a = sa.remove_at(i2);
4314     /*: 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))" */
4315
4316     /*: assume "0 <= i2 & i2 < sb..csize" */
4317     Object r2b = sb.remove_at(i2);
4318     /*: assume "0 <= i1 & i1 < sb..csize" */
4319     Object r1b = sb.remove_at(i1);
4320
4321     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
         sb..csize)" */
4322 }
4323
4324 static void remove_at_remove_at_pre_s_123(ArrayList sa, ArrayList sb, int i1, int
         i2)
4325 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
         sa..contents = sb..contents & sa..csize = sb..csize"
4326     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4327     ensures "True" */
4328 {
4329     /*: 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)" */
4331     /*: assume "0 <= i1 & i1 < sa..csize" */
4332     Object r1a = sa.remove_at(i1);
4333     /*: assume "0 <= i2 & i2 < sa..csize" */
4334     sa.remove_at(i2);
4335
4336     sb.remove_at(i2);
4337     Object r1b = sb.remove_at(i1);
4338
4339     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
4340 }
4341
4342 static void remove_at_remove_at_pre_c_123(ArrayList sa, ArrayList sb, int i1, int
         i2)
4343 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
         sa..contents = sb..contents & sa..csize = sb..csize"
4344     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4345     ensures "True" */
4346 {
4347     /*: 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))" */
4349     /*: assume "0 <= i1 & i1 < sa..csize" */
4350     Object r1a = sa.remove_at(i1);
4351     /*: assume "0 <= i2 & i2 < sa..csize" */
4352     sa.remove_at(i2);
4353
4354     /*: assume "0 <= i2 & i2 < sb..csize" */
4355     sb.remove_at(i2);
4356     /*: assume "0 <= i1 & i1 < sb..csize" */
4357     Object r1b = sb.remove_at(i1);
4358

```

```

4359     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
4360         */
4361 }
4362 static void remove_at_remove_at_between_s_124(ArrayList sa, ArrayList sb, int i1,
4363     int i2)
4364 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4365     sa..contents = sb..contents & sa..csize = sb..csize"
4366     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4367     ensures "True" */
4368 {
4369     /*: assume "0 <= i1 & i1 < sa..csize" */
4370     Object r1a = sa.remove_at(i1);
4371     /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
4372     sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
4373     sa..csize) | (i1 = i2 & (i2, r1a) : sa..contents & 0 <= i2 & i2 < sa..csize)
4374     | (sa..csize > i1 & i1 > i2 & (i1, r1a) : sa..contents & 0 <= i1 & i1 <
4375     sa..csize)" */
4376     /*: assume "0 <= i2 & i2 < sa..csize" */
4377     sa.remove_at(i2);
4378
4379     sb.remove_at(i2);
4380     Object r1b = sb.remove_at(i1);
4381
4382     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
4383 }
4384 static void remove_at_remove_at_between_c_124(ArrayList sa, ArrayList sb, int i1,
4385     int i2)
4386 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4387     sa..contents = sb..contents & sa..csize = sb..csize"
4388     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4389     ensures "True" */
4390 {
4391     /*: assume "0 <= i1 & i1 < sa..csize" */
4392     Object r1a = sa.remove_at(i1);
4393     /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
4394     sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
4395     sa..csize) | (i1 = i2 & (i2, r1a) : sa..contents & 0 <= i2 & i2 < sa..csize)
4396     | (sa..csize > i1 & i1 > i2 & (i1, r1a) : sa..contents & 0 <= i1 & i1 <
4397     sa..csize))" */
4398     /*: assume "0 <= i2 & i2 < sa..csize" */
4399     sa.remove_at(i2);
4400
4401     /*: assume "0 <= i2 & i2 < sb..csize" */
4402     sb.remove_at(i2);
4403     /*: assume "0 <= i1 & i1 < sb..csize" */
4404     Object r1b = sb.remove_at(i1);
4405
4406     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
4407         */
4408 }
4409 static void remove_at_remove_at_post_s_125(ArrayList sa, ArrayList sb, int i1, int
4410     i2)
4411 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4412     sa..contents = sb..contents & sa..csize = sb..csize"
4413     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4414     ensures "True" */
4415 {
4416     /*: assume "0 <= i1 & i1 < sa..csize" */
4417     Object r1a = sa.remove_at(i1);
4418     /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
4419     /*: ghost specvar sa__csize :: "int" = "sa..csize" */
4420     /*: 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 static void remove_at_set_pre_c_126(ArrayList sa, ArrayList sb, int i1, int i2,
4460     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) :
4467     sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) : sa..contents & 0
4468     <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & (ALL
4469     v. ((i1, v) : sa..contents) = ((i2 + 1, v) : sa..contents)) & (i1, v2) :
4470     sa..contents & (i2 + 1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <=
4471     i2 + 1 & i2 + 1 < sa..csize) | (i1 > i2))" */
4472     /*: assume "0 <= i1 & i1 < sa..csize" */
4473     Object r1a = sa.remove_at(i1);
4474     /*: assume "0 <= i2 & i2 < sa..csize" */
4475     Object r2a = sa.set(i2, v2);
4476
4477     /*: assume "0 <= i2 & i2 < sb..csize" */
4478     Object r2b = sb.set(i2, v2);
4479     /*: assume "0 <= i1 & i1 < sb..csize" */
4480     Object r1b = sb.remove_at(i1);
4481
4482     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
4483     sb..csize)" */
4484 }
4485
4486 static void remove_at_set_between_s_127(ArrayList sa, ArrayList sb, int i1, int i2,
4487     Object v2)
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     Object r1a = sa.remove_at(i1);
4495     /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
4496     sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa..contents & 0
4497     <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize) | (i1 = i2 & (i2,
4498     r1a) : sa..contents & r1a = v2 & (i2, v2) : sa..contents & 0 <= i2 & i2 <
4499     sa..csize) | i1 > i2" */
4500     /*: assume "0 <= i2 & i2 < sa..csize" */
4501     Object r2a = sa.set(i2, v2);
4502
4503     Object r2b = sb.set(i2, v2);
4504     Object r1b = sb.remove_at(i1);
4505
4506     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
4507     sb..csize" */
4508 }
4509
4510 static void remove_at_set_between_c_127(ArrayList sa, ArrayList sb, int i1, int i2,
4511     Object v2)
4512 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4513     sa..contents = sb..contents & sa..csize = sb..csize"
4514     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4515     ensures "True" */
4516 {
4517     /*: assume "0 <= i1 & i1 < sa..csize" */
4518     Object r1a = sa.remove_at(i1);
4519     /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
4520     sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa..contents & 0
4521     <= 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)" */
4507 /*: assume "0 <= i2 & i2 < sa..csize" */
4508 Object r2a = sa.set(i2, v2);
4509
4510 /*: assume "0 <= i2 & i2 < sb..csize" */
4511 Object r2b = sb.set(i2, v2);
4512 /*: assume "0 <= i1 & i1 < sb..csize" */
4513 Object r1b = sb.remove_at(i1);
4514
4515 /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
    sb..csize)" */
4516 }
4517
4518 static void remove_at_set_post_s_128(ArrayList sa, ArrayList sb, int i1, int i2,
    Object v2)
4519 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4520     sa..contents = sb..contents & sa..csize = sb..csize"
4521     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4522     ensures "True" */
4523 {
4524     /*: assume "0 <= i1 & i1 < sa..csize" */
4525     Object r1a = sa.remove_at(i1);
4526     /*: assume "0 <= i2 & i2 < sa..csize" */
4527     Object r2a = sa.set(i2, v2);
4528     /*: 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" */
4529
4530     Object r2b = sb.set(i2, v2);
4531     Object r1b = sb.remove_at(i1);
4532
4533     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
4534 }
4535
4536 static void remove_at_set_post_c_128(ArrayList sa, ArrayList sb, int i1, int i2,
    Object v2)
4537 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4538     sa..contents = sb..contents & sa..csize = sb..csize"
4539     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4540     ensures "True" */
4541 {
4542     /*: assume "0 <= i1 & i1 < sa..csize" */
4543     Object r1a = sa.remove_at(i1);
4544     /*: assume "0 <= i2 & i2 < sa..csize" */
4545     Object r2a = sa.set(i2, v2);
4546     /*: 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)" */
4547
4548     /*: assume "0 <= i2 & i2 < sb..csize" */
4549     Object r2b = sb.set(i2, v2);
4550     /*: assume "0 <= i1 & i1 < sb..csize" */
4551     Object r1b = sb.remove_at(i1);
4552
4553     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
4554 }
4555
4556 static void remove_at_set_pre_s_129(ArrayList sa, ArrayList sb, int i1, int i2,
    Object v2)
4557 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4558     sa..contents = sb..contents & sa..csize = sb..csize"
4559     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"

```

```

4560     ensures "True" */
4561 {
4562     /*: 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)" */
4563     /*: assume "0 <= i1 & i1 < sa..csize" */
4564     Object r1a = sa.remove_at(i1);
4565     /*: assume "0 <= i2 & i2 < sa..csize" */
4566     sa.set(i2, v2);
4567
4568     sb.set(i2, v2);
4569     Object r1b = sb.remove_at(i1);
4570
4571     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
4572 }
4573
4574 static void remove_at_set_pre_c_129(ArrayList sa, ArrayList sb, int i1, int i2,
Object v2)
4575 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
sa..contents = sb..contents & sa..csize = sb..csize"
4576     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4577     ensures "True" */
4578 {
4579     /*: 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))" */
4581     /*: assume "0 <= i1 & i1 < sa..csize" */
4582     Object r1a = sa.remove_at(i1);
4583     /*: assume "0 <= i2 & i2 < sa..csize" */
4584     sa.set(i2, v2);
4585
4586     /*: assume "0 <= i2 & i2 < sb..csize" */
4587     sb.set(i2, v2);
4588     /*: assume "0 <= i1 & i1 < sb..csize" */
4589     Object r1b = sb.remove_at(i1);
4590
4591     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
*/
4592 }
4593
4594 static void remove_at_set_between_s_130(ArrayList sa, ArrayList sb, int i1, int i2,
Object v2)
4595 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
sa..contents = sb..contents & sa..csize = sb..csize"
4596     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4597     ensures "True" */
4598 {
4599     /*: assume "0 <= i1 & i1 < sa..csize" */
4600     Object r1a = sa.remove_at(i1);
4601     /*: 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" */
4603     /*: assume "0 <= i2 & i2 < sa..csize" */
4604     sa.set(i2, v2);
4605
4606     sb.set(i2, v2);
4607     Object r1b = sb.remove_at(i1);

```



```

4608     /*: assert "ria = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
4609 }
4610
4611 static void remove_at_set_between_c_130(ArrayList sa, ArrayList sb, int i1, int i2,
4612     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) :
4621     sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa..contents & 0
4622     <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize) | (i1 = i2 & (i2,
4623     r1a) : sa..contents & r1a = v2 & (i2, v2) : sa..contents & 0 <= i2 & i2 <
4624     sa..csize) | i1 > i2)" */
4625     /*: assume "0 <= i2 & i2 < sa..csize" */
4626     sa.set(i2, v2);
4627
4628     /*: assume "0 <= i2 & i2 < sb..csize" */
4629     sb.set(i2, v2);
4630     /*: assume "0 <= i1 & i1 < sb..csize" */
4631     Object r1b = sb.remove_at(i1);
4632
4633     /*: assert "~(ria = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
4634     */
4635 }
4636
4637 static void remove_at_set_post_s_131(ArrayList sa, ArrayList sb, int i1, int i2,
4638     Object v2)
4639 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4640     sa..contents = sb..contents & sa..csize = sb..csize"
4641     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4642     ensures "True" */
4643 {
4644     /*: assume "0 <= i1 & i1 < sa..csize" */
4645     Object r1a = sa.remove_at(i1);
4646     /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
4647     /*: ghost specvar sa__csize :: "int" = "sa..csize" */
4648     /*: assume "0 <= i2 & i2 < sa..csize" */
4649     sa.set(i2, v2);
4650     /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
4651     sa__contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa__contents & 0
4652     <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa__csize) | (i1 = i2 & (i2,
4653     r1a) : sa__contents & r1a = v2 & (i2, v2) : sa__contents & 0 <= i2 & i2 <
4654     sa__csize) | i1 > i2" */
4655
4656     sb.set(i2, v2);
4657     Object r1b = sb.remove_at(i1);
4658
4659     /*: assert "ria = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
4660 }
4661
4662 static void remove_at_set_post_c_131(ArrayList sa, ArrayList sb, int i1, int i2,
4663     Object v2)
4664 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4665     sa..contents = sb..contents & sa..csize = sb..csize"
4666     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4667     ensures "True" */
4668 {
4669     /*: assume "0 <= i1 & i1 < sa..csize" */
4670     Object r1a = sa.remove_at(i1);
4671     /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */

```

```

4661     /*: ghost specvar sa__csize :: "int" = "sa..csize" */
4662     /*: assume "0 <= i2 & i2 < sa..csize" */
4663     sa.set(i2, v2);
4664     /*: 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)" */
4665
4666     /*: assume "0 <= i2 & i2 < sb..csize" */
4667     sb.set(i2, v2);
4668     /*: assume "0 <= i1 & i1 < sb..csize" */
4669     Object r1b = sb.remove_at(i1);
4670
4671     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
         */
4672 }
4673
4674 static void remove_at_size_pre_s_132(ArrayList sa, ArrayList sb, int i1)
4675 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4676     sa..contents = sb..contents & sa..csize = sb..csize"
4677     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4678     ensures "True" */
4679 {
4680     /*: assume "False" */
4681     /*: assume "0 <= i1 & i1 < sa..csize" */
4682     Object r1a = sa.remove_at(i1);
4683     int r2a = sa.size();
4684
4685     int r2b = sb.size();
4686     Object r1b = sb.remove_at(i1);
4687
4688     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
         sb..csize" */
4689 }
4690
4691 static void remove_at_size_pre_c_132(ArrayList sa, ArrayList sb, int i1)
4692 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4693     sa..contents = sb..contents & sa..csize = sb..csize"
4694     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4695     ensures "True" */
4696 {
4697     /*: assume "~(False)" */
4698     /*: assume "0 <= i1 & i1 < sa..csize" */
4699     Object r1a = sa.remove_at(i1);
4700     int r2a = sa.size();
4701
4702     int r2b = sb.size();
4703     /*: assume "0 <= i1 & i1 < sb..csize" */
4704     Object r1b = sb.remove_at(i1);
4705
4706     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
         sb..csize)" */
4707 }
4708
4709 static void remove_at_size_between_s_133(ArrayList sa, ArrayList sb, int i1)
4710 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4711     sa..contents = sb..contents & sa..csize = sb..csize"
4712     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4713     ensures "True" */
4714 {
4715     /*: assume "0 <= i1 & i1 < sa..csize" */
4716     Object r1a = sa.remove_at(i1);
4717     /*: assume "False" */
4718     int r2a = sa.size();

```

```

4719     int r2b = sb.size();
4720     Object r1b = sb.remove_at(i1);
4721
4722     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
4723         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 =
4742         sb..csize)" */
4743 }
4744
4745 static void remove_at_size_post_s_134(ArrayList sa, ArrayList sb, int i1)
4746 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4747     sa..contents = sb..contents & sa..csize = sb..csize"
4748 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4749 ensures "True" */
4750 {
4751     /*: assume "0 <= i1 & i1 < sa..csize" */
4752     Object r1a = sa.remove_at(i1);
4753     int r2a = sa.size();
4754     /*: assume "False" */
4755
4756     int r2b = sb.size();
4757     Object r1b = sb.remove_at(i1);
4758
4759     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
4760         sb..csize" */
4761 }
4762
4763 static void remove_at_size_post_c_134(ArrayList sa, ArrayList sb, int i1)
4764 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4765     sa..contents = sb..contents & sa..csize = sb..csize"
4766 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4767 ensures "True" */
4768 {
4769     /*: assume "0 <= i1 & i1 < sa..csize" */
4770     Object r1a = sa.remove_at(i1);
4771     int r2a = sa.size();
4772     /*: assume "~(False)" */
4773
4774     int r2b = sb.size();
4775     /*: assume "0 <= i1 & i1 < sb..csize" */
4776     Object r1b = sb.remove_at(i1);
4777
4778     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
4779         sb..csize)" */
4780 }

```

```

4779 static void remove_at_add_at_pre_s_135(ArrayList sa, ArrayList sb, int i1, int i2,
4780 Object v2)
4781 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4782 sa..contents = sb..contents & sa..csize = sb..csize"
4783 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
4784 "sb..msize"
4785 ensures "True" */
4786 {
4787 /*: assume "(i1 < i2 & (i2, v2) : sa..contents & 0 <= i2 & i2 < sa..csize) |
4788 (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize) | (i1 > i2 &
4789 (ALL v. ((i1 - 1, v) : sa..contents) = ((i1, v) : sa..contents)) & 0 <= i1 -
4790 1 & i1 - 1 < sa..csize & 0 <= i1 & i1 < sa..csize)" */
4791 /*: assume "0 <= i1 & i1 < sa..csize" */
4792 sa.remove_at(i1);
4793 /*: assume "0 <= i2 & i2 <= sa..csize" */
4794 sa.add_at(i2, v2);
4795
4796 sb.add_at(i2, v2);
4797 sb.remove_at(i1);
4798
4799 /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
4800 }
4801
4802 static void remove_at_add_at_pre_c_135(ArrayList sa, ArrayList sb, int i1, int i2,
4803 Object v2)
4804 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4805 sa..contents = sb..contents & sa..csize = sb..csize"
4806 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
4807 "sb..msize"
4808 ensures "True" */
4809 {
4810 /*: assume "~((i1 < i2 & (i2, v2) : sa..contents & 0 <= i2 & i2 < sa..csize) |
4811 (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize) | (i1 > i2 &
4812 (ALL v. ((i1 - 1, v) : sa..contents) = ((i1, v) : sa..contents)) & 0 <= i1 -
4813 1 & i1 - 1 < sa..csize & 0 <= i1 & i1 < sa..csize))" */
4814 /*: assume "0 <= i1 & i1 < sa..csize" */
4815 sa.remove_at(i1);
4816 /*: assume "0 <= i2 & i2 <= sa..csize" */
4817 sa.add_at(i2, v2);
4818
4819 /*: assume "0 <= i2 & i2 <= sb..csize" */
4820 sb.add_at(i2, v2);
4821 /*: assume "0 <= i1 & i1 < sb..csize" */
4822 sb.remove_at(i1);
4823
4824 /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
4825 }
4826
4827 static void remove_at_add_at_between_s_136(ArrayList sa, ArrayList sb, int i1, int
4828 i2, Object v2)
4829 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4830 sa..contents = sb..contents & sa..csize = sb..csize"
4831 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
4832 "sb..msize"
4833 ensures "True" */
4834 {
4835 /*: assume "0 <= i1 & i1 < sa..csize" */
4836 sa.remove_at(i1);
4837 /*: assume "(i1 < i2 & (i2 - 1, v2) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
4838 sa..csize) | (i1 = i2 & (i1, v2) : sa..(old contents) & 0 <= i1 & i1 <
4839 sa..(old csize)) | (i1 > i2 & (ALL v. ((i1 - 1, v) : sa..contents) = ((i1,
4840 v) : sa..(old contents))) & 0 <= i1 - 1 & i1 - 1 < sa..csize & 0 <= i1 & i1
4841 < sa..(old csize))" */
4842 /*: assume "0 <= i2 & i2 <= sa..csize" */
4843 sa.add_at(i2, v2);

```

```

4828     sb.add_at(i2, v2);
4829     sb.remove_at(i1);
4830
4831     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
4832 }
4833
4834
4835 static void remove_at_add_at_between_c_136(ArrayList sa, ArrayList sb, int i1, int
4836     i2, Object v2)
4837 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4838     sa..contents = sb..contents & sa..csize = sb..csize"
4839     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
4840     "sb..msize"
4841     ensures "True" */
4842 {
4843     /*: assume "0 <= i1 & i1 < sa..csize" */
4844     sa.remove_at(i1);
4845     /*: assume "~((i1 < i2 & (i2 - 1, v2) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
4846     sa..csize) | (i1 = i2 & (i1, v2) : sa..(old contents) & 0 <= i1 & i1 <
4847     sa..(old csize)) | (i1 > i2 & (ALL v. ((i1 - 1, v) : sa..contents) = ((i1,
4848     v) : sa..(old contents))) & 0 <= i1 - 1 & i1 - 1 < sa..csize & 0 <= i1 & i1
4849     < sa..(old csize)))" */
4850     /*: assume "0 <= i2 & i2 <= sa..csize" */
4851     sa.add_at(i2, v2);
4852
4853     /*: assume "0 <= i2 & i2 <= sb..csize" */
4854     sb.add_at(i2, v2);
4855     /*: assume "0 <= i1 & i1 < sb..csize" */
4856     sb.remove_at(i1);
4857
4858     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
4859 }
4860
4861 static void remove_at_add_at_post_s_137(ArrayList sa, ArrayList sb, int i1, int i2,
4862     Object v2)
4863 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4864     sa..contents = sb..contents & sa..csize = sb..csize"
4865     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
4866     "sb..msize"
4867     ensures "True" */
4868 {
4869     /*: assume "0 <= i1 & i1 < sa..csize" */
4870     sa.remove_at(i1);
4871     /*: assume "0 <= i2 & i2 <= sa..csize" */
4872     sa.add_at(i2, v2);
4873     /*: assume "(i1 < i2 & (i2 - 1, v2) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
4874     sa..csize) | (i1 = i2 & (i1, v2) : sa..(old contents) & 0 <= i1 & i1 <
4875     sa..(old csize)) | (i1 > i2 & (ALL v. ((i1, v) : sa..contents) = ((i1, v) :
4876     sa..(old contents))) & 0 <= i1 & i1 < sa..csize & 0 <= i1 & i1 < sa..(old
4877     csize))" */
4878
4879     sb.add_at(i2, v2);
4880     sb.remove_at(i1);
4881
4882     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
4883 }
4884
4885 static void remove_at_add_at_post_c_137(ArrayList sa, ArrayList sb, int i1, int i2,
4886     Object v2)
4887 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4888     sa..contents = sb..contents & sa..csize = sb..csize"
4889     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
4890     "sb..msize"
4891     ensures "True" */
4892 {

```

```

4879     /*: assume "0 <= i1 & i1 < sa..csize" */
4880     sa.remove_at(i1);
4881     /*: assume "0 <= i2 & i2 <= sa..csize" */
4882     sa.add_at(i2, v2);
4883     /*: 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)))" */
4884
4885     /*: assume "0 <= i2 & i2 <= sb..csize" */
4886     sb.add_at(i2, v2);
4887     /*: assume "0 <= i1 & i1 < sb..csize" */
4888     sb.remove_at(i1);
4889
4890     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
4891 }
4892
4893 static void remove_at_get_pre_s_138(ArrayList sa, ArrayList sb, int i1, int i2)
4894 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4895     sa..contents = sb..contents & sa..csize = sb..csize"
4896     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4897     ensures "True" */
4898 {
4899     /*: 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" */
4900     /*: assume "0 <= i1 & i1 < sa..csize" */
4901     sa.remove_at(i1);
4902     /*: assume "0 <= i2 & i2 < sa..csize" */
4903     Object r2a = sa.get(i2);
4904
4905     Object r2b = sb.get(i2);
4906     sb.remove_at(i1);
4907
4908     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
4909 }
4910
4911 static void remove_at_get_pre_c_138(ArrayList sa, ArrayList sb, int i1, int i2)
4912 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4913     sa..contents = sb..contents & sa..csize = sb..csize"
4914     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4915     ensures "True" */
4916 {
4917     /*: 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)" */
4918     /*: assume "0 <= i1 & i1 < sa..csize" */
4919     sa.remove_at(i1);
4920     /*: assume "0 <= i2 & i2 < sa..csize" */
4921     Object r2a = sa.get(i2);
4922
4923     /*: assume "0 <= i2 & i2 < sb..csize" */
4924     Object r2b = sb.get(i2);
4925     /*: assume "0 <= i1 & i1 < sb..csize" */
4926     sb.remove_at(i1);
4927
4928     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
         */
4929 }
4930

```

```

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 <
4998             sa..csize) | (i1 = i2 & (i1, r2a) : sa..(old contents) & 0 <= i1 & i1 <
4999             sa..(old csize)) | i1 > i2)" */
5000
5001     /*: assume "0 <= i2 & i2 < sb..csize" */
5002     Object r2b = sb.get(i2);
5003     /*: assume "0 <= i1 & i1 < sb..csize" */
5004     sb.remove_at(i1);
5005
5006     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
5007             */
5008 }
5009
5010 static void remove_at_indexOf_pre_s_141(ArrayList sa, ArrayList sb, int i1, Object
5011 v2)
5012 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5013             sa..contents = sb..contents & sa..csize = sb..csize"
5014 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5015 ensures "True" */
5016 {
5017     /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
5018             (i, v2) : sa..contents & 0 <= i & i < i1) | ~(EX i. (i, v2) : sa..contents
5019             & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & i1
5020             < sa..csize - 1 & (i1 + 1, v2) : sa..contents & 0 <= i1 + 1 & i1 + 1 <
5021             sa..csize)" */
5022     /*: assume "0 <= i1 & i1 < sa..csize" */
5023     sa.remove_at(i1);
5024     int r2a = sa.indexOf(v2);
5025
5026     int r2b = sb.indexOf(v2);
5027     sb.remove_at(i1);
5028
5029     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
5030 }
5031
5032 static void remove_at_indexOf_pre_c_141(ArrayList sa, ArrayList sb, int i1, Object
5033 v2)
5034 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5035             sa..contents = sb..contents & sa..csize = sb..csize"
5036 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5037 ensures "True" */
5038 {
5039     /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
5040             (i, v2) : sa..contents & 0 <= i & i < i1) | ~(EX i. (i, v2) : sa..contents
5041             & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & i1
5042             < sa..csize - 1 & (i1 + 1, v2) : sa..contents & 0 <= i1 + 1 & i1 + 1 <
5043             sa..csize)" */
5044     /*: assume "0 <= i1 & i1 < sa..csize" */
5045     sa.remove_at(i1);
5046     int r2a = sa.indexOf(v2);
5047
5048     int r2b = sb.indexOf(v2);
5049     /*: assume "0 <= i1 & i1 < sb..csize" */
5050 }

```



```

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     int r2b = sb.indexOf(v2);
5088     sb.remove_at(i1);
5089
5090     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
5091 }
5092
5093
5094 static void remove_at_indexOf_post_c_143(ArrayList sa, ArrayList sb, int i1, Object
5095     v2)
5096 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5097     sa..contents = sb..contents & sa..csize = sb..csize"
5098     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5099     ensures "True" */
5100 {
5101     /*: assume "0 <= i1 & i1 < sa..csize" */
5102     sa.remove_at(i1);
5103     int r2a = sa.indexOf(v2);
5104     /*: assume "~((r2a < 0 & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 <
5105     sa..(old csize)) | (0 <= r2a & r2a < i1) | (r2a = i1 & (i1, v2) : sa..(old
5106     contents) & 0 <= i1 & i1 < sa..(old csize) & i1 < sa..csize))" */
5107
5108     int r2b = sb.indexOf(v2);
5109     /*: assume "0 <= i1 & i1 < sb..csize" */
5110     sb.remove_at(i1);
5111
5112     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
5113     */
5114 }
5115
5116 static void remove_at_lastIndexOf_pre_s_144(ArrayList sa, ArrayList sb, int i1,
5117     Object v2)
5118 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5119     sa..contents = sb..contents & sa..csize = sb..csize"
5120     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5121     ensures "True" */
5122 {
5123     /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | ((EX i.
5124     (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) : sa..contents &
5125     i1 <= i & i < sa..csize))" */
5126     /*: assume "0 <= i1 & i1 < sa..csize" */
5127     sa.remove_at(i1);
5128     int r2a = sa.lastIndexOf(v2);
5129
5130     int r2b = sb.lastIndexOf(v2);
5131     sb.remove_at(i1);
5132
5133     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
5134 }
5135
5136 static void remove_at_lastIndexOf_pre_c_144(ArrayList sa, ArrayList sb, int i1,
5137     Object v2)
5138 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5139     sa..contents = sb..contents & sa..csize = sb..csize"
5140     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5141     ensures "True" */
5142 {
5143     /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | ((EX
5144     i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
5145     sa..contents & i1 <= i & i < sa..csize)))" */
5146     /*: assume "0 <= i1 & i1 < sa..csize" */
5147     sa.remove_at(i1);
5148     int r2a = sa.lastIndexOf(v2);
5149
5150     int r2b = sb.lastIndexOf(v2);
5151     /*: 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
5148 static void remove_at_lastIndexof_between_s_145(ArrayList sa, ArrayList sb, int i1,
5149         Object v2)
5150 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5151     sa..contents = sb..contents & sa..csize = sb..csize"
5152     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5153     ensures "True" */
5154 {
5155     /*: assume "0 <= i1 & i1 < sa..csize" */
5156     sa.remove_at(i1);
5157     /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & (i1,
5158         v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | ((EX i. (i,
5159         v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) : sa..contents & i1
5160         <= i & i < sa..csize) & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 <
5161         sa..(old csize)))" */
5162     int r2a = sa.lastIndexof(v2);
5163
5164     int r2b = sb.lastIndexof(v2);
5165     sb.remove_at(i1);
5166
5167     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
5168 }
5169
5170 static void remove_at_lastIndexof_between_c_145(ArrayList sa, ArrayList sb, int i1,
5171         Object v2)
5172 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5173     sa..contents = sb..contents & sa..csize = sb..csize"
5174     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5175     ensures "True" */
5176 {
5177     /*: assume "0 <= i1 & i1 < sa..csize" */
5178     sa.remove_at(i1);
5179     /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & (i1,
5180         v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | ((EX i. (i,
5181         v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) : sa..contents & i1
5182         <= i & i < sa..csize) & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 <
5183         sa..(old csize)))" */
5184     int r2a = sa.lastIndexof(v2);
5185
5186     int r2b = sb.lastIndexof(v2);
5187     /*: assume "0 <= i1 & i1 < sb..csize" */
5188     sb.remove_at(i1);
5189
5190     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
5191         */
5192 }
5193
5194 static void remove_at_lastIndexof_post_s_146(ArrayList sa, ArrayList sb, int i1,
5195         Object v2)
5196 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5197     sa..contents = sb..contents & sa..csize = sb..csize"
5198     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5199     ensures "True" */
5200 {
5201     /*: assume "0 <= i1 & i1 < sa..csize" */
5202     sa.remove_at(i1);
5203     int r2a = sa.lastIndexof(v2);
5204     /*: assume "(r2a < 0 & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old
5205         csize)) | (0 <= r2a & r2a < i1 & (i1, v2) ~: sa..(old contents) & 0 <= i1 &
5206         i1 < sa..(old csize))" */

```

```

5192     int r2b = sb.lastIndexOf(v2);
5193     sb.remove_at(i1);
5194
5195     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
5196 }
5197
5198
5199 static void remove_at_lastIndexOf_post_c_146(ArrayList sa, ArrayList sb, int i1,
5200     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", "sa..csize", "sb..csize"
5204     ensures "True" */
5205 {
5206     /*: assume "0 <= i1 & i1 < sa..csize" */
5207     sa.remove_at(i1);
5208     int r2a = sa.lastIndexOf(v2);
5209     /*: assume "~((r2a < 0 & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 <
5210     sa..(old csize)) | (0 <= r2a & r2a < i1 & (i1, v2) ~: sa..(old contents) & 0
5211     <= i1 & i1 < sa..(old csize)))" */
5212
5213     int r2b = sb.lastIndexOf(v2);
5214     /*: assume "0 <= i1 & i1 < sb..csize" */
5215     sb.remove_at(i1);
5216
5217     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
5218     */
5219 }
5220
5221 static void remove_at_remove_at_pre_s_147(ArrayList sa, ArrayList sb, int i1, int
5222     i2)
5223 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5224     sa..contents = sb..contents & sa..csize = sb..csize"
5225     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5226     ensures "True" */
5227 {
5228     /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
5229     sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
5230     sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :
5231     sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
5232     sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
5233     sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
5234     <= i1 + 1 & i1 + 1 < sa..csize)" */
5235     /*: assume "0 <= i1 & i1 < sa..csize" */
5236     sa.remove_at(i1);
5237     /*: assume "0 <= i2 & i2 < sa..csize" */
5238     Object r2a = sa.remove_at(i2);
5239
5240     Object r2b = sb.remove_at(i2);
5241     sb.remove_at(i1);
5242
5243     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
5244 }
5245
5246 static void remove_at_remove_at_pre_c_147(ArrayList sa, ArrayList sb, int i1, int
5247     i2)
5248 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5249     sa..contents = sb..contents & sa..csize = sb..csize"
5250     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5251     ensures "True" */
5252 {
5253     /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
5254     sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
5255     sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :
5256     sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <

```

```

5242     sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
5243     sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
5244     <= i1 + 1 & i1 + 1 < sa..csize))" */
5245     /*: assume "0 <= i1 & i1 < sa..csize" */
5246     sa.remove_at(i1);
5247     /*: assume "0 <= i2 & i2 < sa..csize" */
5248     Object r2a = sa.remove_at(i2);
5249     /*: assume "0 <= i2 & i2 < sb..csize" */
5250     Object r2b = sb.remove_at(i2);
5251     /*: assume "0 <= i1 & i1 < sb..csize" */
5252     sb.remove_at(i1);
5253     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
5254     */
5255 }
5256
5257 static void remove_at_remove_at_between_s_148(ArrayList sa, ArrayList sb, int i1,
5258 int i2)
5259 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5260 sa..contents = sb..contents & sa..csize = sb..csize"
5261 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5262 ensures "True" */
5263 {
5264     /*: assume "0 <= i1 & i1 < sa..csize" */
5265     sa.remove_at(i1);
5266     /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
5267     sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
5268     sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..(old contents)) = ((i2, v) :
5269     sa..contents)) & 0 <= i1 & i1 < sa..(old csize) & 0 <= i2 & i2 < sa..csize)
5270     | (sa..csize > i1 & i1 > i2 & (ALL v. ((i1, v) : sa..(old contents)) = ((i1,
5271     v) : sa..contents)) & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 & i1 <
5272     sa..csize)" */
5273     /*: assume "0 <= i2 & i2 < sa..csize" */
5274     Object r2a = sa.remove_at(i2);
5275     Object r2b = sb.remove_at(i2);
5276     sb.remove_at(i1);
5277     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
5278 }
5279
5280 static void remove_at_remove_at_between_c_148(ArrayList sa, ArrayList sb, int i1,
5281 int i2)
5282 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5283 sa..contents = sb..contents & sa..csize = sb..csize"
5284 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5285 ensures "True" */
5286 {
5287     /*: assume "0 <= i1 & i1 < sa..csize" */
5288     sa.remove_at(i1);
5289     /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
5290     sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
5291     sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..(old contents)) = ((i2, v) :
5292     sa..contents)) & 0 <= i1 & i1 < sa..(old csize) & 0 <= i2 & i2 < sa..csize)
5293     | (sa..csize > i1 & i1 > i2 & (ALL v. ((i1, v) : sa..(old contents)) = ((i1,
5294     v) : sa..contents)) & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 & i1 <
5295     sa..csize)" */
5296     /*: assume "0 <= i2 & i2 < sa..csize" */
5297     Object r2a = sa.remove_at(i2);
5298     Object r2b = sb.remove_at(i2);
5299     sb.remove_at(i1);
5300     /*: assume "0 <= i2 & i2 < sb..csize" */
5301     Object r2b = sb.remove_at(i2);
5302     /*: assume "0 <= i1 & i1 < sb..csize" */
5303     sb.remove_at(i1);

```

```

5289     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
5290         */
5291 }
5292
5293 static void remove_at_remove_at_post_s_149(ArrayList sa, ArrayList sb, int i1, int
5294     i2)
5295 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5296     sa..contents = sb..contents & sa..csize = sb..csize"
5297     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5298     ensures "True" */
5299 {
5300     /*: assume "0 <= i1 & i1 < sa..csize" */
5301     sa.remove_at(i1);
5302     /*: assume "0 <= i2 & i2 < sa..csize" */
5303     Object r2a = sa.remove_at(i2);
5304     /*: assume "(i1 < i2 & (i2 - 1, r2a) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
5305         sa..csize) | (i1 = i2 & (i1, r2a) : sa..(old contents) & 0 <= i1 & i1 <
5306         sa..(old csize)) | (sa..csize + 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
5307         sa..(old contents)) = ((i1 - 1, v) : sa..contents)) & 0 <= i1 & i1 <
5308         sa..(old csize) & 0 <= i1 - 1 & i1 - 1 < sa..csize)" */
5309
5310     Object r2b = sb.remove_at(i2);
5311     sb.remove_at(i1);
5312
5313     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
5314 }
5315
5316 static void remove_at_remove_at_post_c_149(ArrayList sa, ArrayList sb, int i1, int
5317     i2)
5318 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5319     sa..contents = sb..contents & sa..csize = sb..csize"
5320     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5321     ensures "True" */
5322 {
5323     /*: assume "0 <= i1 & i1 < sa..csize" */
5324     sa.remove_at(i1);
5325     /*: assume "0 <= i2 & i2 < sa..csize" */
5326     Object r2a = sa.remove_at(i2);
5327     /*: assume "~((i1 < i2 & (i2 - 1, r2a) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
5328         sa..csize) | (i1 = i2 & (i1, r2a) : sa..(old contents) & 0 <= i1 & i1 <
5329         sa..(old csize)) | (sa..csize + 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
5330         sa..(old contents)) = ((i1 - 1, v) : sa..contents)) & 0 <= i1 & i1 <
5331         sa..(old csize) & 0 <= i1 - 1 & i1 - 1 < sa..csize))" */
5332
5333     /*: assume "0 <= i2 & i2 < sb..csize" */
5334     Object r2b = sb.remove_at(i2);
5335     /*: assume "0 <= i1 & i1 < sb..csize" */
5336     sb.remove_at(i1);
5337
5338     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
5339         */
5340 }
5341
5342 static void remove_at_remove_at_pre_s_150(ArrayList sa, ArrayList sb, int i1, int
5343     i2)
5344 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5345     sa..contents = sb..contents & sa..csize = sb..csize"
5346     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5347     ensures "True" */
5348 {
5349     /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
5350         sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
5351         sa..csize) | i1 = i2 | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :

```

```

5338     sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
5339     <= i1 + 1 & i1 + 1 < sa..csize)" */
5340 /*: assume "0 <= i1 & i1 < sa..csize" */
5341 sa.remove_at(i1);
5342 /*: assume "0 <= i2 & i2 < sa..csize" */
5343 sa.remove_at(i2);
5344
5345 sb.remove_at(i2);
5346 sb.remove_at(i1);
5347
5348 /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
5349 }
5350
5351 static void remove_at_remove_at_pre_c_150(ArrayList sa, ArrayList sb, int i1, int
5352 i2)
5353 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5354 sa..contents = sb..contents & sa..csize = sb..csize"
5355 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5356 ensures "True" */
5357 {
5358 /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
5359 sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
5360 sa..csize) | i1 = i2 | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
5361 sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
5362 <= i1 + 1 & i1 + 1 < sa..csize))" */
5363 /*: assume "0 <= i1 & i1 < sa..csize" */
5364 sa.remove_at(i1);
5365 /*: assume "0 <= i2 & i2 < sa..csize" */
5366 sa.remove_at(i2);
5367
5368 /*: assume "0 <= i2 & i2 < sb..csize" */
5369 sb.remove_at(i2);
5370 /*: assume "0 <= i1 & i1 < sb..csize" */
5371 sb.remove_at(i1);
5372
5373 /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
5374 }
5375
5376 static void remove_at_remove_at_between_s_151(ArrayList sa, ArrayList sb, int i1,
5377 int i2)
5378 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5379 sa..contents = sb..contents & sa..csize = sb..csize"
5380 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5381 ensures "True" */
5382 {
5383 /*: assume "0 <= i1 & i1 < sa..csize" */
5384 sa.remove_at(i1);
5385 /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
5386 sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
5387 sa..csize) | i1 = i2 | (sa..csize > i1 & i1 > i2 & (ALL v. ((i1, v) :
5388 sa..(old contents) = ((i1, v) : sa..contents)) & 0 <= i1 & i1 < sa..(old
5389 csize) & 0 <= i1 & i1 < sa..csize)" */
5390 /*: assume "0 <= i2 & i2 < sa..csize" */
5391 sa.remove_at(i2);
5392
5393 sb.remove_at(i2);
5394 sb.remove_at(i1);
5395
5396 /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
5397 }
5398
5399 static void remove_at_remove_at_between_c_151(ArrayList sa, ArrayList sb, int i1,
5400 int i2)
5401 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5402 sa..contents = sb..contents & sa..csize = sb..csize"

```

```

5390     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5391     ensures "True" */
5392 {
5393     /*: assume "0 <= i1 & i1 < sa..csize" */
5394     sa.remove_at(i1);
5395     /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
5396         sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
5397         sa..csize) | i1 = i2 | (sa..csize > i1 & i1 > i2 & (ALL v. ((i1, v) :
5398         sa..(old contents)) = ((i1, v) : sa..contents)) & 0 <= i1 & i1 < sa..(old
5399         csize) & 0 <= i1 & i1 < sa..csize))" */
5400     /*: assume "0 <= i2 & i2 < sa..csize" */
5401     sa.remove_at(i2);
5402
5403     /*: assume "0 <= i2 & i2 < sb..csize" */
5404     sb.remove_at(i2);
5405     /*: assume "0 <= i1 & i1 < sb..csize" */
5406     sb.remove_at(i1);
5407
5408     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
5409 }
5410
5411 static void remove_at_remove_at_post_s_152(ArrayList sa, ArrayList sb, int i1, int
5412     i2)
5413 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5414     sa..contents = sb..contents & sa..csize = sb..csize"
5415 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5416 ensures "True" */
5417 {
5418     /*: assume "0 <= i1 & i1 < sa..csize" */
5419     sa.remove_at(i1);
5420     /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
5421     /*: ghost specvar sa__csize :: "int" = "sa..csize" */
5422     /*: assume "0 <= i2 & i2 < sa..csize" */
5423     sa.remove_at(i2);
5424     /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
5425         sa__contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
5426         sa__csize) | i1 = i2 | (sa..csize + 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
5427         sa..(old contents)) = ((i1 - 1, v) : sa..contents)) & 0 <= i1 & i1 <
5428         sa..(old csize) & 0 <= i1 - 1 & i1 - 1 < sa..csize)" */
5429
5430     sb.remove_at(i2);
5431     sb.remove_at(i1);
5432
5433     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
5434 }
5435
5436 static void remove_at_remove_at_post_c_152(ArrayList sa, ArrayList sb, int i1, int
5437     i2)
5438 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5439     sa..contents = sb..contents & sa..csize = sb..csize"
5440 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5441 ensures "True" */
5442 {
5443     /*: assume "0 <= i1 & i1 < sa..csize" */
5444     sa.remove_at(i1);
5445     /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
5446     /*: ghost specvar sa__csize :: "int" = "sa..csize" */
5447     /*: assume "0 <= i2 & i2 < sa..csize" */
5448     sa.remove_at(i2);
5449     /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
5450         sa__contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
5451         sa__csize) | i1 = i2 | (sa..csize + 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
5452         sa..(old contents)) = ((i1 - 1, v) : sa..contents)) & 0 <= i1 & i1 <
5453         sa..(old csize) & 0 <= i1 - 1 & i1 - 1 < sa..csize))" */
5454

```



```

5441     /*: assume "0 <= i2 & i2 < sb..csize" */
5442     sb.remove_at(i2);
5443     /*: assume "0 <= i1 & i1 < sb..csize" */
5444     sb.remove_at(i1);
5445
5446     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
5447 }
5448
5449 static void remove_at_set_pre_s_153(ArrayList sa, ArrayList sb, int i1, int i2,
5450     Object v2)
5451 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5452     sa..contents = sb..contents & sa..csize = sb..csize"
5453     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5454     ensures "True" */
5455 {
5456     /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
5457     sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) : sa..contents & 0
5458     <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & (ALL
5459     v. ((i1, v) : sa..contents) = ((i2 + 1, v) : sa..contents)) & (i1, v2) :
5460     sa..contents & (i2 + 1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <=
5461     i2 + 1 & i2 + 1 < sa..csize) | (i1 > i2)" */
5462     /*: assume "0 <= i1 & i1 < sa..csize" */
5463     sa.remove_at(i1);
5464     /*: assume "0 <= i2 & i2 < sa..csize" */
5465     Object r2a = sa.set(i2, v2);
5466
5467     Object r2b = sb.set(i2, v2);
5468     sb.remove_at(i1);
5469
5470     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
5471 }
5472
5473 static void remove_at_set_pre_c_153(ArrayList sa, ArrayList sb, int i1, int i2,
5474     Object v2)
5475 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5476     sa..contents = sb..contents & sa..csize = sb..csize"
5477     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5478     ensures "True" */
5479 {
5480     /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
5481     sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) : sa..contents & 0
5482     <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & (ALL
5483     v. ((i1, v) : sa..contents) = ((i2 + 1, v) : sa..contents)) & (i1, v2) :
5484     sa..contents & (i2 + 1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <=
5485     i2 + 1 & i2 + 1 < sa..csize) | (i1 > i2))" */
5486     /*: assume "0 <= i1 & i1 < sa..csize" */
5487     sa.remove_at(i1);
5488     /*: assume "0 <= i2 & i2 < sa..csize" */
5489     Object r2a = sa.set(i2, v2);
5490
5491     /*: assume "0 <= i2 & i2 < sb..csize" */
5492     Object r2b = sb.set(i2, v2);
5493     /*: assume "0 <= i1 & i1 < sb..csize" */
5494     sb.remove_at(i1);
5495
5496     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
5497     */
5498 }
5499
5500 static void remove_at_set_between_s_154(ArrayList sa, ArrayList sb, int i1, int i2,
5501     Object v2)
5502 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5503     sa..contents = sb..contents & sa..csize = sb..csize"
5504     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5505     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,
5544   Object v2)
5545 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5546   sa..contents = sb..contents & sa..csize = sb..csize"
5547   modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5548   ensures "True" */
5549 {
5550   /*: assume "0 <= i1 & i1 < sa..csize" */
5551   sa.remove_at(i1);
5552   /*: assume "0 <= i2 & i2 < sa..csize" */
5553   Object r2a = sa.set(i2, v2);
5554   /*: assume "~((i1 < i2 & (i2 - 1, r2a) : sa..contents & (i2 - 1, v2) :
5555     sa..contents & r2a = v2 & 0 <= i2 - 1 & i2 - 1 < sa..csize) | (i1 = i2 &
5556     (i1, r2a) : sa..(old contents) & (i1, v2) : sa..(old contents) & r2a = v2 &
5557     0 <= i1 & i1 < sa..(old csize)) | i1 > i2)" */
5558
5559   /*: assume "0 <= i2 & i2 < sb..csize" */
5560   Object r2b = sb.set(i2, v2);
5561   /*: assume "0 <= i1 & i1 < sb..csize" */
5562   sb.remove_at(i1);
5563
5564   /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
5565     */
5566 }
5567
5568 static void remove_at_set_pre_s_156(ArrayList sa, ArrayList sb, int i1, int i2,
5569   Object v2)
5570 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5571   sa..contents = sb..contents & sa..csize = sb..csize"
5572   modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5573   ensures "True" */
5574 {
5575   /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
5576     sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) : sa..contents & 0
5577     <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & (i2
5578     + 1, v2) : sa..contents & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 > i2)" */
5579   /*: assume "0 <= i1 & i1 < sa..csize" */
5580   sa.remove_at(i1);
5581   /*: assume "0 <= i2 & i2 < sa..csize" */
5582   sa.set(i2, v2);
5583
5584   sb.set(i2, v2);
5585   sb.remove_at(i1);
5586
5587   /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
5588 }
5589
5590 static void remove_at_set_pre_c_156(ArrayList sa, ArrayList sb, int i1, int i2,
5591   Object v2)
5592 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5593   sa..contents = sb..contents & sa..csize = sb..csize"
5594   modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5595   ensures "True" */
5596 {
5597   /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
5598     sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) : sa..contents & 0
5599     <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & (i2
5600     + 1, v2) : sa..contents & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 > i2))" */
5601   /*: assume "0 <= i1 & i1 < sa..csize" */
5602   sa.remove_at(i1);
5603   /*: assume "0 <= i2 & i2 < sa..csize" */
5604   sa.set(i2, v2);
5605 }

```

```

5593     /*: assume "0 <= i2 & i2 < sb..csize" */
5594     sb.set(i2, v2);
5595     /*: assume "0 <= i1 & i1 < sb..csize" */
5596     sb.remove_at(i1);
5597
5598     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
5599 }
5600
5601 static void remove_at_set_between_s_157(ArrayList sa, ArrayList sb, int i1, int i2,
5602     Object v2)
5603 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5604     sa..contents = sb..contents & sa..csize = sb..csize"
5605     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5606     ensures "True" */
5607 {
5608     /*: assume "0 <= i1 & i1 < sa..csize" */
5609     sa.remove_at(i1);
5610     /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
5611     sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa..contents & 0
5612     <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize) | (i1 = i2 & (i2,
5613     v2) : sa..contents & 0 <= i2 & i2 < sa..csize) | i1 > i2" */
5614     /*: assume "0 <= i2 & i2 < sa..csize" */
5615     sa.set(i2, v2);
5616
5617     sb.set(i2, v2);
5618     sb.remove_at(i1);
5619
5620     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
5621 }
5622
5623 static void remove_at_set_between_c_157(ArrayList sa, ArrayList sb, int i1, int i2,
5624     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", "sa..csize", "sb..csize"
5628     ensures "True" */
5629 {
5630     /*: assume "0 <= i1 & i1 < sa..csize" */
5631     sa.remove_at(i1);
5632     /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
5633     sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa..contents & 0
5634     <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize) | (i1 = i2 & (i2,
5635     v2) : sa..contents & 0 <= i2 & i2 < sa..csize) | i1 > i2)" */
5636     /*: assume "0 <= i2 & i2 < sa..csize" */
5637     sa.set(i2, v2);
5638
5639     /*: assume "0 <= i2 & i2 < sb..csize" */
5640     sb.set(i2, v2);
5641     /*: assume "0 <= i1 & i1 < sb..csize" */
5642     sb.remove_at(i1);
5643
5644     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
5645 }
5646
5647 static void remove_at_set_post_s_158(ArrayList sa, ArrayList sb, int i1, int i2,
5648     Object v2)
5649 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5650     sa..contents = sb..contents & sa..csize = sb..csize"
5651     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5652     ensures "True" */
5653 {
5654     /*: assume "0 <= i1 & i1 < sa..csize" */
5655     sa.remove_at(i1);
5656     /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
5657     /*: 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);

```

```

5707     int r2a = sa.size();
5708
5709     int r2b = sb.size();
5710     /*: assume "0 <= i1 & i1 < sb..csize" */
5711     sb.remove_at(i1);
5712
5713     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
5714 }
5715
5716 static void remove_at_size_between_s_160(ArrayList sa, ArrayList sb, int i1)
5717 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5718     sa..contents = sb..contents & sa..csize = sb..csize"
5719 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5720 ensures "True" */
5721 {
5722     /*: assume "0 <= i1 & i1 < sa..csize" */
5723     sa.remove_at(i1);
5724     /*: assume "False" */
5725     int r2a = sa.size();
5726
5727     int r2b = sb.size();
5728     sb.remove_at(i1);
5729
5730     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
5731 }
5732
5733 static void remove_at_size_between_c_160(ArrayList sa, ArrayList sb, int i1)
5734 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5735     sa..contents = sb..contents & sa..csize = sb..csize"
5736 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5737 ensures "True" */
5738 {
5739     /*: assume "0 <= i1 & i1 < sa..csize" */
5740     sa.remove_at(i1);
5741     /*: assume "~(False)" */
5742     int r2a = sa.size();
5743
5744     int r2b = sb.size();
5745     /*: assume "0 <= i1 & i1 < sb..csize" */
5746     sb.remove_at(i1);
5747
5748     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
5749 }
5750
5751 static void remove_at_size_post_s_161(ArrayList sa, ArrayList sb, int i1)
5752 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5753     sa..contents = sb..contents & sa..csize = sb..csize"
5754 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5755 ensures "True" */
5756 {
5757     /*: assume "0 <= i1 & i1 < sa..csize" */
5758     sa.remove_at(i1);
5759     int r2a = sa.size();
5760     /*: assume "False" */
5761
5762     int r2b = sb.size();
5763     sb.remove_at(i1);
5764
5765     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
5766 }
5767
5768 static void remove_at_size_post_c_161(ArrayList sa, ArrayList sb, int i1)
5769 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &

```

```

5770         sa..contents = sb..contents & sa..csize = sb..csize"
5771 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5772 ensures "True" */
5773 {
5774     /*: assume "0 <= i1 & i1 < sa..csize" */
5775     sa.remove_at(i1);
5776     int r2a = sa.size();
5777     /*: assume "~(False)" */
5778
5779     int r2b = sb.size();
5780     /*: assume "0 <= i1 & i1 < sb..csize" */
5781     sb.remove_at(i1);
5782
5783     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
5784         */
5785 }
5786
5787 static void set_add_at_pre_s_162(ArrayList sa, ArrayList sb, int i1, Object v1, int
5788 i2, Object v2)
5789 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5790     sa..contents = sb..contents & sa..csize = sb..csize"
5791 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
5792     "sb..msize"
5793 ensures "True" */
5794 {
5795     /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
5796     sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | (i1 > i2 & (ALL v. ((i1
5797     - 1, v) : sa..contents) = ((i1, v) : sa..contents)) & (i1 - 1, v1) :
5798     sa..contents & (i1, v1) : sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize &
5799     0 <= i1 & i1 < sa..csize)" */
5800     /*: assume "0 <= i1 & i1 < sa..csize" */
5801     Object r1a = sa.set(i1, v1);
5802     /*: assume "0 <= i2 & i2 <= sa..csize" */
5803     sa.add_at(i2, v2);
5804
5805     sb.add_at(i2, v2);
5806     Object r1b = sb.set(i1, v1);
5807
5808     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
5809 }
5810
5811 static void set_add_at_pre_c_162(ArrayList sa, ArrayList sb, int i1, Object v1, int
5812 i2, Object v2)
5813 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5814     sa..contents = sb..contents & sa..csize = sb..csize"
5815 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
5816     "sb..msize"
5817 ensures "True" */
5818 {
5819     /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
5820     sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | (i1 > i2 & (ALL v. ((i1
5821     - 1, v) : sa..contents) = ((i1, v) : sa..contents)) & (i1 - 1, v1) :
5822     sa..contents & (i1, v1) : sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize &
5823     0 <= i1 & i1 < sa..csize))" */
5824     /*: assume "0 <= i1 & i1 < sa..csize" */
5825     Object r1a = sa.set(i1, v1);
5826     /*: assume "0 <= i2 & i2 <= sa..csize" */
5827     sa.add_at(i2, v2);
5828
5829     /*: assume "0 <= i2 & i2 <= sb..csize" */
5830     sb.add_at(i2, v2);
5831     /*: assume "0 <= i1 & i1 < sb..csize" */
5832     Object r1b = sb.set(i1, v1);
5833 }

```

```

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,
5825     int i2, Object v2)
5826 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5827     sa..contents = sb..contents & sa..csize = sb..csize"
5828     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
5829     "sb..msize"
5830     ensures "True" */
5831 {
5832     /*: assume "0 <= i1 & i1 < sa..csize" */
5833     Object r1a = sa.set(i1, v1);
5834     /*: assume "i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | (i1 > i2 &
5835     (i1 - 1, r1a) : sa..contents & (i1 - 1, v1) : sa..contents & r1a = v1 & 0 <=
5836     i1 - 1 & i1 - 1 < sa..csize)" */
5837     /*: assume "0 <= i2 & i2 <= sa..csize" */
5838     sa.add_at(i2, v2);
5839     sb.add_at(i2, v2);
5840     Object r1b = sb.set(i1, v1);
5841     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
5842 }
5843 static void set_add_at_between_c_163(ArrayList sa, ArrayList sb, int i1, Object v1,
5844     int i2, Object v2)
5845 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5846     sa..contents = sb..contents & sa..csize = sb..csize"
5847     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
5848     "sb..msize"
5849     ensures "True" */
5850 {
5851     /*: assume "0 <= i1 & i1 < sa..csize" */
5852     Object r1a = sa.set(i1, v1);
5853     /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | (i1 > i2 &
5854     (i1 - 1, r1a) : sa..contents & (i1 - 1, v1) : sa..contents & r1a = v1 & 0 <=
5855     i1 - 1 & i1 - 1 < sa..csize))" */
5856     /*: assume "0 <= i2 & i2 <= sa..csize" */
5857     sa.add_at(i2, v2);
5858     sb.add_at(i2, v2);
5859     /*: assume "0 <= i1 & i1 < sb..csize" */
5860     Object r1b = sb.set(i1, v1);
5861     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
5862         */
5863 }
5864 static void set_add_at_post_s_164(ArrayList sa, ArrayList sb, int i1, Object v1,
5865     int i2, Object v2)
5866 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5867     sa..contents = sb..contents & sa..csize = sb..csize"
5868     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
5869     "sb..msize"
5870     ensures "True" */
5871 {
5872     /*: assume "0 <= i1 & i1 < sa..csize" */
5873     Object r1a = sa.set(i1, v1);
5874     /*: assume "0 <= i2 & i2 <= sa..csize" */
5875     sa.add_at(i2, v2);

```



```

5872     /*: assume "i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | (i1 > i2 &
5873         (i1, r1a) : sa..contents & (i1, v1) : sa..contents & r1a = v1 & 0 <= i1 & i1
5874         < sa..csize)" */
5875     sb.add_at(i2, v2);
5876     Object r1b = sb.set(i1, v1);
5877
5878     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
5879 }
5880
5881 static void set_add_at_post_c_164(ArrayList sa, ArrayList sb, int i1, Object v1,
5882     int i2, Object v2)
5883 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5884     sa..contents = sb..contents & sa..csize = sb..csize"
5885     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
5886     "sb..msize"
5887     ensures "True" */
5888 {
5889     /*: assume "0 <= i1 & i1 < sa..csize" */
5890     Object r1a = sa.set(i1, v1);
5891     /*: assume "0 <= i2 & i2 <= sa..csize" */
5892     sa.add_at(i2, v2);
5893     /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | (i1 > i2 &
5894         (i1, r1a) : sa..contents & (i1, v1) : sa..contents & r1a = v1 & 0 <= i1 & i1
5895         < sa..csize))" */
5896
5897     /*: assume "0 <= i2 & i2 <= sb..csize" */
5898     sb.add_at(i2, v2);
5899     /*: assume "0 <= i1 & i1 < sb..csize" */
5900     Object r1b = sb.set(i1, v1);
5901
5902     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
5903         */
5904 }
5905
5906 static void set_get_pre_s_165(ArrayList sa, ArrayList sb, int i1, Object v1, int i2)
5907 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5908     sa..contents = sb..contents & sa..csize = sb..csize"
5909     modifies "sa..contents", "sb..contents"
5910     ensures "True" */
5911 {
5912     /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & 0 <= i1 & i1 <
5913         sa..csize) | i1 > i2" */
5914     /*: assume "0 <= i1 & i1 < sa..csize" */
5915     Object r1a = sa.set(i1, v1);
5916     /*: assume "0 <= i2 & i2 < sa..csize" */
5917     Object r2a = sa.get(i2);
5918
5919     Object r2b = sb.get(i2);
5920     Object r1b = sb.set(i1, v1);
5921
5922     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
5923         sb..csize" */
5924 }
5925
5926 static void set_get_pre_c_165(ArrayList sa, ArrayList sb, int i1, Object v1, int i2)
5927 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5928     sa..contents = sb..contents & sa..csize = sb..csize"
5929     modifies "sa..contents", "sb..contents"
5930     ensures "True" */
5931 {
5932     /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & 0 <= i1 & i1 <
5933         sa..csize) | i1 > i2)" */
5934     /*: assume "0 <= i1 & i1 < sa..csize" */
5935     Object r1a = sa.set(i1, v1);

```

```

5927     /*: assume "0 <= i2 & i2 < sa..csize" */
5928     Object r2a = sa.get(i2);
5929
5930     /*: assume "0 <= i2 & i2 < sb..csize" */
5931     Object r2b = sb.get(i2);
5932     /*: assume "0 <= i1 & i1 < sb..csize" */
5933     Object r1b = sb.set(i1, v1);
5934
5935     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
5936 }
5937
5938 static void set_get_between_s_166(ArrayList sa, ArrayList sb, int i1, Object v1,
    int i2)
5939 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5940             sa..contents = sb..contents & sa..csize = sb..csize"
5941     modifies "sa..contents", "sb..contents"
5942     ensures "True" */
5943 {
5944     /*: assume "0 <= i1 & i1 < sa..csize" */
5945     Object r1a = sa.set(i1, v1);
5946     /*: assume "i1 < i2 | (i1 = i2 & r1a = v1) | i1 > i2" */
5947     /*: assume "0 <= i2 & i2 < sa..csize" */
5948     Object r2a = sa.get(i2);
5949
5950     Object r2b = sb.get(i2);
5951     Object r1b = sb.set(i1, v1);
5952
5953     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
5954 }
5955
5956 static void set_get_between_c_166(ArrayList sa, ArrayList sb, int i1, Object v1,
    int i2)
5957 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5958             sa..contents = sb..contents & sa..csize = sb..csize"
5959     modifies "sa..contents", "sb..contents"
5960     ensures "True" */
5961 {
5962     /*: assume "0 <= i1 & i1 < sa..csize" */
5963     Object r1a = sa.set(i1, v1);
5964     /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v1) | i1 > i2)" */
5965     /*: assume "0 <= i2 & i2 < sa..csize" */
5966     Object r2a = sa.get(i2);
5967
5968     /*: assume "0 <= i2 & i2 < sb..csize" */
5969     Object r2b = sb.get(i2);
5970     /*: assume "0 <= i1 & i1 < sb..csize" */
5971     Object r1b = sb.set(i1, v1);
5972
5973     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
5974 }
5975
5976 static void set_get_post_s_167(ArrayList sa, ArrayList sb, int i1, Object v1, int
    i2)
5977 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5978             sa..contents = sb..contents & sa..csize = sb..csize"
5979     modifies "sa..contents", "sb..contents"
5980     ensures "True" */
5981 {
5982     /*: assume "0 <= i1 & i1 < sa..csize" */
5983     Object r1a = sa.set(i1, v1);
5984     /*: assume "0 <= i2 & i2 < sa..csize" */
5985     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))" */
6038 /*: assume "0 <= i1 & i1 < sa..csize" */
6039 Object r1a = sa.set(i1, v1);
6040 int r2a = sa.indexOf(v2);
6041
6042 int r2b = sb.indexOf(v2);
6043 /*: assume "0 <= i1 & i1 < sb..csize" */
6044 Object r1b = sb.set(i1, v1);
6045
6046 /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
6047 }
6048
6049 static void set_indexOf_between_s_169(ArrayList sa, ArrayList sb, int i1, Object
        v1, Object v2)
6050 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
6051 modifies "sa..contents", "sb..contents"
6052 ensures "True" */
6053 {
6054     /*: assume "0 <= i1 & i1 < sa..csize" */
6055     Object r1a = sa.set(i1, v1);
6056     /*: 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)" */
6057     int r2a = sa.indexOf(v2);
6058
6059     int r2b = sb.indexOf(v2);
6060     Object r1b = sb.set(i1, v1);
6061
6062     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
6063 }
6064
6065 static void set_indexOf_between_c_169(ArrayList sa, ArrayList sb, int i1, Object
        v1, Object v2)
6066 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
6067 modifies "sa..contents", "sb..contents"
6068 ensures "True" */
6069 {
6070     /*: assume "0 <= i1 & i1 < sa..csize" */
6071     Object r1a = sa.set(i1, v1);
6072     /*: 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))" */
6073     int r2a = sa.indexOf(v2);
6074
6075     int r2b = sb.indexOf(v2);
6076     /*: assume "0 <= i1 & i1 < sb..csize" */
6077     Object r1b = sb.set(i1, v1);
6078
6079     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
6080 }
6081
6082 static void set_indexOf_post_s_170(ArrayList sa, ArrayList sb, int i1, Object v1,
        Object v2)
6083 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
6084
6085
6086

```

```

6087     modifies "sa..contents", "sb..contents"
6088     ensures "True" */
6089
6090     {
6091         /*: assume "0 <= i1 & i1 < sa..csize" */
6092         Object r1a = sa.set(i1, v1);
6093         int r2a = sa.indexOf(v2);
6094         /*: assume "(r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1) | (r2a = i1 & r1a =
6095            v2) | (r2a > i1 & r1a ~= v2)" */
6096
6097         int r2b = sb.indexOf(v2);
6098         Object r1b = sb.set(i1, v1);
6099
6100         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
6101            sb..csize" */
6102     }
6103
6104     static void set_indexOf_post_c_170(ArrayList sa, ArrayList sb, int i1, Object v1,
6105         Object v2)
6106     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6107        sa..contents = sb..contents & sa..csize = sb..csize"
6108        modifies "sa..contents", "sb..contents"
6109        ensures "True" */
6110     {
6111         /*: assume "0 <= i1 & i1 < sa..csize" */
6112         Object r1a = sa.set(i1, v1);
6113         int r2a = sa.indexOf(v2);
6114         /*: assume "~((r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1) | (r2a = i1 & r1a =
6115            v2) | (r2a > i1 & r1a ~= v2))" */
6116
6117         int r2b = sb.indexOf(v2);
6118         /*: assume "0 <= i1 & i1 < sb..csize" */
6119         Object r1b = sb.set(i1, v1);
6120
6121         /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
6122            sb..csize)" */
6123     }
6124
6125     static void set_lastIndexOf_pre_s_171(ArrayList sa, ArrayList sb, int i1, Object
6126         v1, Object v2)
6127     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6128        sa..contents = sb..contents & sa..csize = sb..csize"
6129        modifies "sa..contents", "sb..contents"
6130        ensures "True" */
6131     {
6132         /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
6133            v2) | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
6134            sa..contents & i1 <= i & i < sa..csize) & v1 ~= v2) | ((i1, v2) :
6135            sa..contents & 0 <= i1 & i1 < sa..csize & ~(EX i. (i, v2) : sa..contents &
6136            i1 < i & i < sa..csize) & v1 = v2) | (EX i. (i, v2) : sa..contents & i1 < i
6137            & i < sa..csize)" */
6138         /*: assume "0 <= i1 & i1 < sa..csize" */
6139         Object r1a = sa.set(i1, v1);
6140         int r2a = sa.lastIndexOf(v2);
6141
6142         int r2b = sb.lastIndexOf(v2);
6143         Object r1b = sb.set(i1, v1);
6144
6145         /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
6146            sb..csize" */
6147     }
6148
6149     static void set_lastIndexOf_pre_c_171(ArrayList sa, ArrayList sb, int i1, Object
6150         v1, Object v2)
6151     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6152        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 &
        sa..contents = sb..contents & sa..csize = sb..csize"
6156     modifies "sa..contents", "sb..contents"
6157     ensures "True" */
6158 {
6159     /*: assume "0 <= i1 & i1 < sa..csize" */
6160     Object r1a = sa.set(i1, v1);
6161     /*: 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)" */
6162     int r2a = sa.lastIndexOf(v2);
6163
6164     int r2b = sb.lastIndexOf(v2);
6165     Object r1b = sb.set(i1, v1);
6166
6167     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
6168 }
6169
6170 static void set_lastIndexOf_between_c_172(ArrayList sa, ArrayList sb, int i1,
        Object v1, Object v2)
6171 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
6172     modifies "sa..contents", "sb..contents"
6173     ensures "True" */
6174 {
6175     /*: assume "0 <= i1 & i1 < sa..csize" */
6176     Object r1a = sa.set(i1, v1);
6177     /*: 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))" */
6178     int r2a = sa.lastIndexOf(v2);
6179
6180     int r2b = sb.lastIndexOf(v2);
6181     /*: assume "0 <= i1 & i1 < sb..csize" */
6182     Object r1b = sb.set(i1, v1);
6183
6184

```

```

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 &
        sa..contents = sb..contents & sa..csize = sb..csize"
6191     modifies "sa..contents", "sb..contents"
6192     ensures "True" */
6193 {
6194     /*: assume "0 <= i1 & i1 < sa..csize" */
6195     Object r1a = sa.set(i1, v1);
6196     int r2a = sa.lastIndexOf(v2);
6197     /*: assume "(r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1 & r1a ~= v2) | (r2a =
        i1 & r1a = v2) | r2a > i1" */
6198
6199     int r2b = sb.lastIndexOf(v2);
6200     Object r1b = sb.set(i1, v1);
6201
6202     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
6203 }
6204
6205 static void set_lastIndexOf_post_c_173(ArrayList sa, ArrayList sb, int i1, Object
        v1, Object v2)
6206 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
6207     modifies "sa..contents", "sb..contents"
6208     ensures "True" */
6209 {
6210     /*: assume "0 <= i1 & i1 < sa..csize" */
6211     Object r1a = sa.set(i1, v1);
6212     int r2a = sa.lastIndexOf(v2);
6213     /*: assume "~((r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1 & r1a ~= v2) | (r2a
        = i1 & r1a = v2) | r2a > i1)" */
6214
6215     int r2b = sb.lastIndexOf(v2);
6216     /*: assume "0 <= i1 & i1 < sb..csize" */
6217     Object r1b = sb.set(i1, v1);
6218
6219     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
6220 }
6221
6222 static void set_remove_at_pre_s_174(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2)
6223 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
6224     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
6225     ensures "True" */
6226 {
6227     /*: 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)" */
6228
6229     /*: assume "0 <= i1 & i1 < sa..csize" */
6230     Object r1a = sa.set(i1, v1);
6231     /*: assume "0 <= i2 & i2 < sa..csize" */
6232     Object r2a = sa.remove_at(i2);
6233
6234
6235

```

```

6236     Object r2b = sb.remove_at(i2);
6237     Object r1b = sb.set(i1, v1);
6238
6239     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
6240 }
6241
6242 static void set_remove_at_pre_c_174(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2)
6243 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6244     sa..contents = sb..contents & sa..csize = sb..csize"
6245     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
6246     ensures "True" */
6247 {
6248     /*: 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))" */
6249     /*: assume "0 <= i1 & i1 < sa..csize" */
6250     Object r1a = sa.set(i1, v1);
6251     /*: assume "0 <= i2 & i2 < sa..csize" */
6252     Object r2a = sa.remove_at(i2);
6253
6254     /*: assume "0 <= i2 & i2 < sb..csize" */
6255     Object r2b = sb.remove_at(i2);
6256     /*: assume "0 <= i1 & i1 < sb..csize" */
6257     Object r1b = sb.set(i1, v1);
6258
6259     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
6260 }
6261
6262 static void set_remove_at_between_s_175(ArrayList sa, ArrayList sb, int i1, Object
        v1, int i2)
6263 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6264     sa..contents = sb..contents & sa..csize = sb..csize"
6265     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
6266     ensures "True" */
6267 {
6268     /*: assume "0 <= i1 & i1 < sa..csize" */
6269     Object r1a = sa.set(i1, v1);
6270     /*: 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)" */
6271     /*: assume "0 <= i2 & i2 < sa..csize" */
6272     Object r2a = sa.remove_at(i2);
6273
6274     Object r2b = sb.remove_at(i2);
6275     Object r1b = sb.set(i1, v1);
6276
6277     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
6278 }
6279
6280 static void set_remove_at_between_c_175(ArrayList sa, ArrayList sb, int i1, Object
        v1, int i2)
6281 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6282     sa..contents = sb..contents & sa..csize = sb..csize"
6283     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
6284     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 =
6336         sb..csize)" */
6337 }
6338
6339 static void set_remove_at_pre_s_177(ArrayList sa, ArrayList sb, int i1, Object v1,
6340     int i2)
6341 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6342     sa..contents = sb..contents & sa..csize = sb..csize"
6343     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
6344     ensures "True" */
6345 {
6346     /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (ALL v. ((i1, v) :
6347         sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
6348         (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
6349         1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
6350         sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
6351         (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
6352         1 < sa..csize)" */
6353     /*: assume "0 <= i1 & i1 < sa..csize" */
6354     Object r1a = sa.set(i1, v1);
6355     /*: assume "0 <= i2 & i2 < sa..csize" */
6356     sa.remove_at(i2);
6357
6358     sb.remove_at(i2);
6359     Object r1b = sb.set(i1, v1);
6360
6361     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
6362 }
6363
6364 static void set_remove_at_pre_c_177(ArrayList sa, ArrayList sb, int i1, Object v1,
6365     int i2)
6366 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6367     sa..contents = sb..contents & sa..csize = sb..csize"
6368     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
6369     ensures "True" */
6370 {
6371     /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (ALL v. ((i1, v) :
6372         sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
6373         (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
6374         1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
6375         sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
6376         (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
6377         1 < sa..csize))" */
6378     /*: assume "0 <= i1 & i1 < sa..csize" */
6379     Object r1a = sa.set(i1, v1);
6380     /*: assume "0 <= i2 & i2 < sa..csize" */
6381     sa.remove_at(i2);
6382
6383     /*: assume "0 <= i2 & i2 < sb..csize" */
6384     sb.remove_at(i2);
6385     /*: assume "0 <= i1 & i1 < sb..csize" */
6386     Object r1b = sb.set(i1, v1);
6387
6388     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
6389         */
6390 }
6391
6392 static void set_remove_at_between_s_178(ArrayList sa, ArrayList sb, int i1, Object
6393     v1, int i2)
6394 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6395     sa..contents = sb..contents & sa..csize = sb..csize"
6396     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
6397     ensures "True" */
6398 {
6399     /*: 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)

```

```

6433  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6434          sa..contents = sb..contents & sa..csize = sb..csize"
6435  modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
6436  ensures "True" */
6437  {
6438      /*: assume "0 <= i1 & i1 < sa..csize" */
6439      Object r1a = sa.set(i1, v1);
6440      /*: assume "0 <= i2 & i2 < sa..csize" */
6441      sa.remove_at(i2);
6442      /*: 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))" */
6443
6444      /*: assume "0 <= i2 & i2 < sb..csize" */
6445      sb.remove_at(i2);
6446      /*: assume "0 <= i1 & i1 < sb..csize" */
6447      Object r1b = sb.set(i1, v1);
6448
6449      /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
6450  }
6451
6452  static void set_set_pre_s_180(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
6453  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6454          sa..contents = sb..contents & sa..csize = sb..csize"
6455  modifies "sa..contents", "sb..contents"
6456  ensures "True" */
6457  {
6458      /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
        sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | i1 > i2" */
6459      /*: assume "0 <= i1 & i1 < sa..csize" */
6460      Object r1a = sa.set(i1, v1);
6461      /*: assume "0 <= i2 & i2 < sa..csize" */
6462      Object r2a = sa.set(i2, v2);
6463
6464      Object r2b = sb.set(i2, v2);
6465      Object r1b = sb.set(i1, v1);
6466
6467      /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
6468  }
6469
6470  static void set_set_pre_c_180(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
6471  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6472          sa..contents = sb..contents & sa..csize = sb..csize"
6473  modifies "sa..contents", "sb..contents"
6474  ensures "True" */
6475  {
6476      /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
        sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | i1 > i2)" */
6477      /*: assume "0 <= i1 & i1 < sa..csize" */
6478      Object r1a = sa.set(i1, v1);
6479      /*: assume "0 <= i2 & i2 < sa..csize" */
6480      Object r2a = sa.set(i2, v2);
6481
6482      /*: assume "0 <= i2 & i2 < sb..csize" */
6483      Object r2b = sb.set(i2, v2);
6484      /*: assume "0 <= i1 & i1 < sb..csize" */
6485      Object r1b = sb.set(i1, v1);
6486
6487      /*: 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,
6491     int i2, Object v2)
6492 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6493     sa..contents = sb..contents & sa..csize = sb..csize"
6494     modifies "sa..contents", "sb..contents"
6495     ensures "True" */
6496 {
6497     /*: assume "0 <= i1 & i1 < sa..csize" */
6498     Object r1a = sa.set(i1, v1);
6499     /*: assume "i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | i1 > i2" */
6500     /*: assume "0 <= i2 & i2 < sa..csize" */
6501     Object r2a = sa.set(i2, v2);
6502
6503     Object r2b = sb.set(i2, v2);
6504     Object r1b = sb.set(i1, v1);
6505
6506     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
6507         sb..csize" */
6508 }
6509
6510 static void set_set_between_c_181(ArrayList sa, ArrayList sb, int i1, Object v1,
6511     int i2, Object v2)
6512 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6513     sa..contents = sb..contents & sa..csize = sb..csize"
6514     modifies "sa..contents", "sb..contents"
6515     ensures "True" */
6516 {
6517     /*: assume "0 <= i1 & i1 < sa..csize" */
6518     Object r1a = sa.set(i1, v1);
6519     /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | i1 > i2)" */
6520     /*: assume "0 <= i2 & i2 < sa..csize" */
6521     Object r2a = sa.set(i2, v2);
6522
6523     /*: assume "0 <= i2 & i2 < sb..csize" */
6524     Object r2b = sb.set(i2, v2);
6525     /*: assume "0 <= i1 & i1 < sb..csize" */
6526     Object r1b = sb.set(i1, v1);
6527
6528     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
6529         sb..csize)" */
6530 }
6531
6532 static void set_set_post_s_182(ArrayList sa, ArrayList sb, int i1, Object v1, int
6533     i2, Object v2)
6534 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6535     sa..contents = sb..contents & sa..csize = sb..csize"
6536     modifies "sa..contents", "sb..contents"
6537     ensures "True" */
6538 {
6539     /*: assume "0 <= i1 & i1 < sa..csize" */
6540     Object r1a = sa.set(i1, v1);
6541     /*: assume "0 <= i2 & i2 < sa..csize" */
6542     Object r2a = sa.set(i2, v2);
6543     /*: assume "i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | i1 > i2" */
6544
6545     Object r2b = sb.set(i2, v2);
6546     Object r1b = sb.set(i1, v1);
6547
6548     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
6549         sb..csize" */
6550 }

```

```

6546 static void set_set_post_c_182(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
6547 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6548         sa..contents = sb..contents & sa..csize = sb..csize"
6549     modifies "sa..contents", "sb..contents"
6550     ensures "True" */
6551 {
6552     /*: assume "0 <= i1 & i1 < sa..csize" */
6553     Object r1a = sa.set(i1, v1);
6554     /*: assume "0 <= i2 & i2 < sa..csize" */
6555     Object r2a = sa.set(i2, v2);
6556     /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | i1 > i2)" */
6557
6558     /*: assume "0 <= i2 & i2 < sb..csize" */
6559     Object r2b = sb.set(i2, v2);
6560     /*: assume "0 <= i1 & i1 < sb..csize" */
6561     Object r1b = sb.set(i1, v1);
6562
6563     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
6564 }
6565
6566 static void set_set_pre_s_183(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
6567 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6568         sa..contents = sb..contents & sa..csize = sb..csize"
6569     modifies "sa..contents", "sb..contents"
6570     ensures "True" */
6571 {
6572     /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
        sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | i1 > i2" */
6573     /*: assume "0 <= i1 & i1 < sa..csize" */
6574     Object r1a = sa.set(i1, v1);
6575     /*: assume "0 <= i2 & i2 < sa..csize" */
6576     sa.set(i2, v2);
6577
6578     sb.set(i2, v2);
6579     Object r1b = sb.set(i1, v1);
6580
6581     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
6582 }
6583
6584 static void set_set_pre_c_183(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
6585 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6586         sa..contents = sb..contents & sa..csize = sb..csize"
6587     modifies "sa..contents", "sb..contents"
6588     ensures "True" */
6589 {
6590     /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
        sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | i1 > i2)" */
6591     /*: assume "0 <= i1 & i1 < sa..csize" */
6592     Object r1a = sa.set(i1, v1);
6593     /*: assume "0 <= i2 & i2 < sa..csize" */
6594     sa.set(i2, v2);
6595
6596     /*: assume "0 <= i2 & i2 < sb..csize" */
6597     sb.set(i2, v2);
6598     /*: assume "0 <= i1 & i1 < sb..csize" */
6599     Object r1b = sb.set(i1, v1);
6600
6601     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
6602 }
6603

```

```

6604 static void set_set_between_s_184(ArrayList sa, ArrayList sb, int i1, Object v1,
6605     int i2, Object v2)
6606 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6607     sa..contents = sb..contents & sa..csize = sb..csize"
6608     modifies "sa..contents", "sb..contents"
6609     ensures "True" */
6610 {
6611     /*: assume "0 <= i1 & i1 < sa..csize" */
6612     Object r1a = sa.set(i1, v1);
6613     /*: assume "i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | i1 > i2" */
6614     /*: assume "0 <= i2 & i2 < sa..csize" */
6615     sa.set(i2, v2);
6616
6617     sb.set(i2, v2);
6618     Object r1b = sb.set(i1, v1);
6619
6620     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
6621 }
6622
6623 static void set_set_between_c_184(ArrayList sa, ArrayList sb, int i1, Object v1,
6624     int i2, Object v2)
6625 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6626     sa..contents = sb..contents & sa..csize = sb..csize"
6627     modifies "sa..contents", "sb..contents"
6628     ensures "True" */
6629 {
6630     /*: assume "0 <= i1 & i1 < sa..csize" */
6631     Object r1a = sa.set(i1, v1);
6632     /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | i1 > i2)" */
6633     /*: assume "0 <= i2 & i2 < sa..csize" */
6634     sa.set(i2, v2);
6635
6636     /*: assume "0 <= i2 & i2 < sb..csize" */
6637     sb.set(i2, v2);
6638     /*: assume "0 <= i1 & i1 < sb..csize" */
6639     Object r1b = sb.set(i1, v1);
6640
6641     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
6642     */
6643 }
6644
6645 static void set_set_post_s_185(ArrayList sa, ArrayList sb, int i1, Object v1, int
6646     i2, Object v2)
6647 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6648     sa..contents = sb..contents & sa..csize = sb..csize"
6649     modifies "sa..contents", "sb..contents"
6650     ensures "True" */
6651 {
6652     /*: assume "0 <= i1 & i1 < sa..csize" */
6653     Object r1a = sa.set(i1, v1);
6654     /*: assume "0 <= i2 & i2 < sa..csize" */
6655     sa.set(i2, v2);
6656     /*: assume "i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | i1 > i2" */
6657
6658     sb.set(i2, v2);
6659     Object r1b = sb.set(i1, v1);
6660
6661     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
6662 }
6663
6664 static void set_set_post_c_185(ArrayList sa, ArrayList sb, int i1, Object v1, int
6665     i2, Object v2)
6666 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6667     sa..contents = sb..contents & sa..csize = sb..csize"
6668     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)

```

```

6786  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6787          sa..contents = sb..contents & sa..csize = sb..csize"
6788  modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
        "sb..msize"
6789  ensures "True" */
6790  {
6791    /*: 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)" */
6792    /*: assume "0 <= i1 & i1 < sa..csize" */
6793    sa.set(i1, v1);
6794    /*: assume "0 <= i2 & i2 <= sa..csize" */
6795    sa.add_at(i2, v2);
6796
6797    sb.add_at(i2, v2);
6798    sb.set(i1, v1);
6799
6800    /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
6801  }
6802
6803  static void set_add_at_pre_c_189(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2, Object v2)
6804  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
6805  modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
        "sb..msize"
6806  ensures "True" */
6807  {
6808    /*: 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))" */
6809    /*: assume "0 <= i1 & i1 < sa..csize" */
6810    sa.set(i1, v1);
6811    /*: assume "0 <= i2 & i2 <= sa..csize" */
6812    sa.add_at(i2, v2);
6813
6814
6815    /*: assume "0 <= i2 & i2 <= sb..csize" */
6816    sb.add_at(i2, v2);
6817    /*: assume "0 <= i1 & i1 < sb..csize" */
6818    sb.set(i1, v1);
6819
6820    /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
6821  }
6822
6823  static void set_add_at_between_s_190(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2, Object v2)
6824  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
6825  modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
        "sb..msize"
6826  ensures "True" */
6827  {
6828    /*: assume "0 <= i1 & i1 < sa..csize" */
6829    sa.set(i1, v1);
6830    /*: 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))" */
6831    /*: assume "0 <= i2 & i2 <= sa..csize" */
6832    sa.add_at(i2, v2);
6833

```

```

6834     sb.add_at(i2, v2);
6835     sb.set(i1, v1);
6836
6837     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
6838 }
6839
6840
6841 static void set_add_at_between_c_190(ArrayList sa, ArrayList sb, int i1, Object v1,
6842     int i2, Object v2)
6843 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6844     sa..contents = sb..contents & sa..csize = sb..csize"
6845     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
6846     "sb..msize"
6847     ensures "True" */
6848 {
6849     /*: assume "0 <= i1 & i1 < sa..csize" */
6850     sa.set(i1, v1);
6851     /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & (i1, v2) :
6852     sa..(old contents) & v1 = v2 & 0 <= i1 & i1 < sa..(old csize)) | (i1 > i2 &
6853     (ALL v. ((i1 - 1, v) : sa..contents) = ((i1, v) : sa..(old contents))) & (i1
6854     - 1, v1) : sa..contents & (i1, v1) : sa..(old contents) & 0 <= i1 - 1 & i1 -
6855     1 < sa..csize & 0 <= i1 & i1 < sa..(old csize)))" */
6856     /*: assume "0 <= i2 & i2 <= sa..csize" */
6857     sa.add_at(i2, v2);
6858
6859     /*: assume "0 <= i2 & i2 <= sb..csize" */
6860     sb.add_at(i2, v2);
6861     /*: assume "0 <= i1 & i1 < sb..csize" */
6862     sb.set(i1, v1);
6863
6864     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
6865 }
6866
6867 static void set_add_at_post_s_191(ArrayList sa, ArrayList sb, int i1, Object v1,
6868     int i2, Object v2)
6869 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6870     sa..contents = sb..contents & sa..csize = sb..csize"
6871     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
6872     "sb..msize"
6873     ensures "True" */
6874 {
6875     /*: assume "0 <= i1 & i1 < sa..csize" */
6876     sa.set(i1, v1);
6877     /*: assume "0 <= i2 & i2 <= sa..csize" */
6878     sa.add_at(i2, v2);
6879     /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & (i1, v2) :
6880     sa..(old contents) & v1 = v2 & 0 <= i1 & i1 < sa..(old csize)) | (i1 > i2 &
6881     (ALL v. ((i1, v) : sa..contents) = ((i1, v) : sa..(old contents))) & (i1, v1)
6882     : sa..contents & (i1, v1) : sa..(old contents) & 0 <= i1 & i1 < sa..csize &
6883     0 <= i1 & i1 < sa..(old csize)))" */
6884     sb.add_at(i2, v2);
6885     sb.set(i1, v1);
6886
6887     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
6888 }
6889
6890 static void set_add_at_post_c_191(ArrayList sa, ArrayList sb, int i1, Object v1,
6891     int i2, Object v2)
6892 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6893     sa..contents = sb..contents & sa..csize = sb..csize"
6894     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
6895     "sb..msize"
6896     ensures "True" */
6897 {

```

```

6885     /*: assume "0 <= i1 & i1 < sa..csize" */
6886     sa.set(i1, v1);
6887     /*: assume "0 <= i2 & i2 <= sa..csize" */
6888     sa.add_at(i2, v2);
6889     /*: 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)))" */
6890
6891     /*: assume "0 <= i2 & i2 <= sb..csize" */
6892     sb.add_at(i2, v2);
6893     /*: assume "0 <= i1 & i1 < sb..csize" */
6894     sb.set(i1, v1);
6895
6896     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
6897 }
6898
6899 static void set_get_pre_s_192(ArrayList sa, ArrayList sb, int i1, Object v1, int i2)
6900 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6901     sa..contents = sb..contents & sa..csize = sb..csize"
6902     modifies "sa..contents", "sb..contents"
6903     ensures "True" */
6904 {
6905     /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & 0 <= i1 & i1 <
        sa..csize) | i1 > i2" */
6906     /*: assume "0 <= i1 & i1 < sa..csize" */
6907     sa.set(i1, v1);
6908     /*: assume "0 <= i2 & i2 < sa..csize" */
6909     Object r2a = sa.get(i2);
6910
6911     Object r2b = sb.get(i2);
6912     sb.set(i1, v1);
6913
6914     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
6915 }
6916
6917 static void set_get_pre_c_192(ArrayList sa, ArrayList sb, int i1, Object v1, int i2)
6918 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6919     sa..contents = sb..contents & sa..csize = sb..csize"
6920     modifies "sa..contents", "sb..contents"
6921     ensures "True" */
6922 {
6923     /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & 0 <= i1 & i1 <
        sa..csize) | i1 > i2)" */
6924     /*: assume "0 <= i1 & i1 < sa..csize" */
6925     sa.set(i1, v1);
6926     /*: assume "0 <= i2 & i2 < sa..csize" */
6927     Object r2a = sa.get(i2);
6928
6929     /*: assume "0 <= i2 & i2 < sb..csize" */
6930     Object r2b = sb.get(i2);
6931     /*: assume "0 <= i1 & i1 < sb..csize" */
6932     sb.set(i1, v1);
6933
6934     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
6935 }
6936
6937 static void set_get_between_s_193(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2)
6938 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6939     sa..contents = sb..contents & sa..csize = sb..csize"
6940     modifies "sa..contents", "sb..contents"
6941     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 &
        sa..contents = sb..contents & sa..csize = sb..csize"
6957     modifies "sa..contents", "sb..contents"
6958     ensures "True" */
6959 {
6960     /*: assume "0 <= i1 & i1 < sa..csize" */
6961     sa.set(i1, v1);
6962     /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & 0 <= i1 & i1
        < sa..(old csize)) | i1 > i2)" */
6963     /*: assume "0 <= i2 & i2 < sa..csize" */
6964     Object r2a = sa.get(i2);
6965
6966     /*: assume "0 <= i2 & i2 < sb..csize" */
6967     Object r2b = sb.get(i2);
6968     /*: assume "0 <= i1 & i1 < sb..csize" */
6969     sb.set(i1, v1);
6970
6971     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
6972 }
6973
6974 static void set_get_post_s_194(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2)
6975 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
6976     modifies "sa..contents", "sb..contents"
6977     ensures "True" */
6978 {
6979     /*: assume "0 <= i1 & i1 < sa..csize" */
6980     sa.set(i1, v1);
6981     /*: assume "0 <= i2 & i2 < sa..csize" */
6982     Object r2a = sa.get(i2);
6983     /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & 0 <= i1 & i1 <
        sa..(old csize)) | i1 > i2" */
6984
6985     Object r2b = sb.get(i2);
6986     sb.set(i1, v1);
6987
6988     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
6989 }
6990
6991 static void set_get_post_c_194(ArrayList sa, ArrayList sb, int i1, Object v1, int
        i2)
6992 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
6993     modifies "sa..contents", "sb..contents"
6994     ensures "True" */
6995 {
6996     /*: assume "0 <= i1 & i1 < sa..csize" */
6997
6998
6999

```

```

7000     sa.set(i1, v1);
7001     /*: assume "0 <= i2 & i2 < sa..csize" */
7002     Object r2a = sa.get(i2);
7003     /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & 0 <= i1 & i1
       < sa..(old csize)) | i1 > i2)" */
7004
7005     /*: assume "0 <= i2 & i2 < sb..csize" */
7006     Object r2b = sb.get(i2);
7007     /*: assume "0 <= i1 & i1 < sb..csize" */
7008     sb.set(i1, v1);
7009
7010     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
       */
7011 }
7012
7013 static void set_indexOf_pre_s_195(ArrayList sa, ArrayList sb, int i1, Object v1,
       Object v2)
7014 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
       sa..contents = sb..contents & sa..csize = sb..csize"
7015 modifies "sa..contents", "sb..contents"
7016 ensures "True" */
7017 {
7018     /*: 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)" */
7019     /*: assume "0 <= i1 & i1 < sa..csize" */
7020     sa.set(i1, v1);
7021     int r2a = sa.indexOf(v2);
7022
7023     int r2b = sb.indexOf(v2);
7024     sb.set(i1, v1);
7025
7026     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
7027 }
7028
7029 static void set_indexOf_pre_c_195(ArrayList sa, ArrayList sb, int i1, Object v1,
       Object v2)
7030 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
       sa..contents = sb..contents & sa..csize = sb..csize"
7031 modifies "sa..contents", "sb..contents"
7032 ensures "True" */
7033 {
7034     /*: 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))" */
7035     /*: assume "0 <= i1 & i1 < sa..csize" */
7036     sa.set(i1, v1);
7037     int r2a = sa.indexOf(v2);
7038
7039     int r2b = sb.indexOf(v2);
7040     sb.set(i1, v1);
7041
7042     /*: assume "0 <= i1 & i1 < sb..csize" */
7043     sb.set(i1, v1);
7044
7045     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
       */
7046 }
7047
7048 static void set_indexOf_between_s_196(ArrayList sa, ArrayList sb, int i1, Object
       v1, Object v2)
7049 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
       sa..contents = sb..contents & sa..csize = sb..csize"
7050

```

```

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 &
       sa..contents = sb..contents & sa..csize = sb..csize"
7067     modifies "sa..contents", "sb..contents"
7068     ensures "True" */
7069 {
7070     /*: assume "0 <= i1 & i1 < sa..csize" */
7071     sa.set(i1, v1);
7072     /*: 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)))" */
7073     int r2a = sa.indexOf(v2);
7074
7075     int r2b = sb.indexOf(v2);
7076     /*: assume "0 <= i1 & i1 < sb..csize" */
7077     sb.set(i1, v1);
7078
7079     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
       */
7080 }
7081
7082 static void set_indexOf_post_s_197(ArrayList sa, ArrayList sb, int i1, Object v1,
       Object v2)
7083 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
       sa..contents = sb..contents & sa..csize = sb..csize"
7084     modifies "sa..contents", "sb..contents"
7085     ensures "True" */
7086 {
7087     /*: assume "0 <= i1 & i1 < sa..csize" */
7088     sa.set(i1, v1);
7089     int r2a = sa.indexOf(v2);
7090     /*: 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))" */
7091
7092     int r2b = sb.indexOf(v2);
7093     sb.set(i1, v1);
7094
7095

```

```

7096     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
7097 }
7098
7099
7100 static void set_indexOf_post_c_197(ArrayList sa, ArrayList sb, int i1, Object v1,
7101     Object v2)
7102 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7103     sa..contents = sb..contents & sa..csize = sb..csize"
7104     modifies "sa..contents", "sb..contents"
7105     ensures "True" */
7106 {
7107     /*: assume "0 <= i1 & i1 < sa..csize" */
7108     sa.set(i1, v1);
7109     int r2a = sa.indexOf(v2);
7110     /*: assume "~((r2a < 0 & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 <
7111         sa..(old csize) | (0 <= r2a & r2a < i1) | (r2a = i1 & (i1, v2) : sa..(old
7112         contents) & 0 <= i1 & i1 < sa..(old csize)) | (r2a > i1 & (i1, v2) ~:
7113         sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)))" */
7114
7115     int r2b = sb.indexOf(v2);
7116     /*: assume "0 <= i1 & i1 < sb..csize" */
7117     sb.set(i1, v1);
7118
7119     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
7120     */
7121 }
7122
7123 static void set_lastIndexOf_pre_s_198(ArrayList sa, ArrayList sb, int i1, Object
7124     v1, Object v2)
7125 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7126     sa..contents = sb..contents & sa..csize = sb..csize"
7127     modifies "sa..contents", "sb..contents"
7128     ensures "True" */
7129 {
7130     /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
7131         v2 | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
7132         sa..contents & i1 <= i & i < sa..csize) & v1 ~= v2) | ((i1, v2) :
7133         sa..contents & 0 <= i1 & i1 < sa..csize & ~(EX i. (i, v2) : sa..contents &
7134         i1 < i & i < sa..csize) & v1 = v2) | (EX i. (i, v2) : sa..contents & i1 < i
7135         & i < sa..csize)" */
7136     /*: assume "0 <= i1 & i1 < sa..csize" */
7137     sa.set(i1, v1);
7138     int r2a = sa.lastIndexOf(v2);
7139
7140     int r2b = sb.lastIndexOf(v2);
7141     sb.set(i1, v1);
7142
7143     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
7144 }
7145
7146 static void set_lastIndexOf_pre_c_198(ArrayList sa, ArrayList sb, int i1, Object
7147     v1, Object v2)
7148 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7149     sa..contents = sb..contents & sa..csize = sb..csize"
7150     modifies "sa..contents", "sb..contents"
7151     ensures "True" */
7152 {
7153     /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
7154         v2 | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
7155         sa..contents & i1 <= i & i < sa..csize) & v1 ~= v2) | ((i1, v2) :
7156         sa..contents & 0 <= i1 & i1 < sa..csize & ~(EX i. (i, v2) : sa..contents &
7157         i1 < i & i < sa..csize) & v1 = v2) | (EX i. (i, v2) : sa..contents & i1 < i
7158         & i < sa..csize)" */
7159     /*: assume "0 <= i1 & i1 < sa..csize" */
7160     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,
7154     Object v1, Object v2)
7155 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7156     sa..contents = sb..contents & sa..csize = sb..csize"
7157     modifies "sa..contents", "sb..contents"
7158     ensures "True" */
7159 {
7160     /*: assume "0 <= i1 & i1 < sa..csize" */
7161     sa.set(i1, v1);
7162     /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & (i1,
7163     v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | ((EX i. (i,
7164     v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) : sa..contents & i1
7165     <= i & i < sa..csize) & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 <
7166     sa..(old csize)) | ((i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize &
7167     ~(EX i. (i, v2) : sa..contents & i1 < i & i < sa..csize) & (i1, v2) :
7168     sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | (EX i. (i, v2) :
7169     sa..contents & i1 < i & i < sa..csize))" */
7170     int r2a = sa.lastIndexOf(v2);
7171
7172     int r2b = sb.lastIndexOf(v2);
7173     sb.set(i1, v1);
7174
7175     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
7176 }
7177
7178 static void set_lastIndexOf_between_c_199(ArrayList sa, ArrayList sb, int i1,
7179     Object v1, Object v2)
7180 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7181     sa..contents = sb..contents & sa..csize = sb..csize"
7182     modifies "sa..contents", "sb..contents"
7183     ensures "True" */
7184 {
7185     /*: assume "0 <= i1 & i1 < sa..csize" */
7186     sa.set(i1, v1);
7187     /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & (i1,
7188     v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | ((EX i. (i,
7189     v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) : sa..contents & i1
7190     <= i & i < sa..csize) & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 <
7191     sa..(old csize)) | ((i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize &
7192     ~(EX i. (i, v2) : sa..contents & i1 < i & i < sa..csize) & (i1, v2) :
7193     sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | (EX i. (i, v2) :
7194     sa..contents & i1 < i & i < sa..csize))" */
7195     int r2a = sa.lastIndexOf(v2);
7196
7197     int r2b = sb.lastIndexOf(v2);
7198     /*: assume "0 <= i1 & i1 < sb..csize" */
7199     sb.set(i1, v1);
7200
7201     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
7202 }
7203
7204 static void set_lastIndexOf_post_s_200(ArrayList sa, ArrayList sb, int i1, Object
7205     v1, Object v2)
7206 /*: 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
7198         csize)) | (0 <= r2a & r2a < i1 & (i1, v2) ~: sa..(old contents) & 0 <= i1 &
7199         i1 < sa..(old csize)) | (r2a = i1 & (i1, v2) : sa..(old contents) & 0 <= i1
7200         & i1 < sa..(old csize)) | r2a > i1" */
7201
7202     int r2b = sb.lastIndexOf(v2);
7203     sb.set(i1, v1);
7204
7205     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
7206 }
7207
7208 static void set_lastIndexOf_post_c_200(ArrayList sa, ArrayList sb, int i1, Object
7209     v1, Object v2)
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"
7213 ensures "True" */
7214 {
7215     /*: assume "0 <= i1 & i1 < sa..csize" */
7216     sa.set(i1, v1);
7217     int r2a = sa.lastIndexOf(v2);
7218     /*: assume "~((r2a < 0 & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 <
7219         sa..(old csize)) | (0 <= r2a & r2a < i1 & (i1, v2) ~: sa..(old contents) & 0
7220         <= i1 & i1 < sa..(old csize)) | (r2a = i1 & (i1, v2) : sa..(old contents) &
7221         0 <= i1 & i1 < sa..(old csize)) | r2a > i1)" */
7222
7223     int r2b = sb.lastIndexOf(v2);
7224     /*: assume "0 <= i1 & i1 < sb..csize" */
7225     sb.set(i1, v1);
7226
7227     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
7228         */
7229 }
7230
7231 static void set_remove_at_pre_s_201(ArrayList sa, ArrayList sb, int i1, Object v1,
7232     int i2)
7233 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7234     sa..contents = sb..contents & sa..csize = sb..csize"
7235 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7236 ensures "True" */
7237 {
7238     /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (ALL v. ((i1, v) :
7239         sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
7240         (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
7241         1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
7242         sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
7243         (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
7244         1 < sa..csize)" */
7245     /*: assume "0 <= i1 & i1 < sa..csize" */
7246     sa.set(i1, v1);
7247     /*: assume "0 <= i2 & i2 < sa..csize" */
7248     Object r2a = sa.remove_at(i2);
7249
7250     Object r2b = sb.remove_at(i2);
7251     sb.set(i1, v1);
7252
7253     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
7254 }

```

```

7240 static void set_remove_at_pre_c_201(ArrayList sa, ArrayList sb, int i1, Object v1,
7241     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) :
7248     sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
7249     (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
7250     1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
7251     sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
7252     (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
7253     1 < sa..csize))" */
7254     /*: assume "0 <= i1 & i1 < sa..csize" */
7255     sa.set(i1, v1);
7256     /*: assume "0 <= i2 & i2 < sa..csize" */
7257     Object r2a = sa.remove_at(i2);
7258     /*: assume "0 <= i2 & i2 < sb..csize" */
7259     Object r2b = sb.remove_at(i2);
7260     /*: assume "0 <= i1 & i1 < sb..csize" */
7261     sb.set(i1, v1);
7262     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
7263     */
7264 }
7265
7266 static void set_remove_at_between_s_202(ArrayList sa, ArrayList sb, int i1, Object
7267     v1, int i2)
7268 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7269     sa..contents = sb..contents & sa..csize = sb..csize"
7270     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7271     ensures "True" */
7272 {
7273     /*: assume "0 <= i1 & i1 < sa..csize" */
7274     sa.set(i1, v1);
7275     /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (ALL v. ((i1, v) :
7276     sa..(old contents)) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..(old
7277     contents) & (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) &
7278     0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v.
7279     ((i1, v) : sa..(old contents)) = ((i1 + 1, v) : sa..contents)) & (i1, v1) :
7280     sa..(old contents) & (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old
7281     csize) & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
7282     /*: assume "0 <= i2 & i2 < sa..csize" */
7283     Object r2a = sa.remove_at(i2);
7284     Object r2b = sb.remove_at(i2);
7285     sb.set(i1, v1);
7286     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
7287 }
7288
7289 static void set_remove_at_between_c_202(ArrayList sa, ArrayList sb, int i1, Object
7290     v1, int i2)
7291 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7292     sa..contents = sb..contents & sa..csize = sb..csize"
7293     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7294     ensures "True" */
7295 {
7296     /*: assume "0 <= i1 & i1 < sa..csize" */
7297     sa.set(i1, v1);
7298     /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (ALL v. ((i1, v) :
7299     sa..(old contents)) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..(old

```

```

7288         contents) & (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) & 0
7289         <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v.
7290         ((i1, v) : sa..(old contents)) = ((i1 + 1, v) : sa..contents)) & (i1, v1) :
7291         sa..(old contents) & (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old
7292         csize) & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
7293     /*: assume "0 <= i2 & i2 < sa..csize" */
7294     Object r2a = sa.remove_at(i2);
7295
7296     /*: assume "0 <= i2 & i2 < sb..csize" */
7297     Object r2b = sb.remove_at(i2);
7298     /*: assume "0 <= i1 & i1 < sb..csize" */
7299     sb.set(i1, v1);
7300
7301     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
7302     */
7303 }
7304
7305 static void set_remove_at_post_s_203(ArrayList sa, ArrayList sb, int i1, Object v1,
7306 int i2)
7307 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7308 sa..contents = sb..contents & sa..csize = sb..csize"
7309 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7310 ensures "True" */
7311 {
7312     /*: assume "0 <= i1 & i1 < sa..csize" */
7313     sa.set(i1, v1);
7314     /*: assume "0 <= i2 & i2 < sa..csize" */
7315     Object r2a = sa.remove_at(i2);
7316     /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize & (ALL v. ((i1, v) : sa..(old
7317     contents)) = ((i1, v) : sa..contents)) & (i1, v1) : sa..(old contents) &
7318     (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 & i1 <
7319     sa..csize) | (sa..csize > i1 & i1 > i2 & (ALL v. ((i1, v) : sa..(old
7320     contents)) = ((i1, v) : sa..contents)) & (i1, v1) : sa..(old contents) &
7321     (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 & i1 <
7322     sa..csize)" */
7323
7324     Object r2b = sb.remove_at(i2);
7325     sb.set(i1, v1);
7326
7327     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
7328 }
7329
7330 static void set_remove_at_post_c_203(ArrayList sa, ArrayList sb, int i1, Object v1,
7331 int i2)
7332 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7333 sa..contents = sb..contents & sa..csize = sb..csize"
7334 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7335 ensures "True" */
7336 {
7337     /*: assume "0 <= i1 & i1 < sa..csize" */
7338     sa.set(i1, v1);
7339     /*: assume "0 <= i2 & i2 < sa..csize" */
7340     Object r2a = sa.remove_at(i2);
7341     /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize & (ALL v. ((i1, v) : sa..(old
7342     contents)) = ((i1, v) : sa..contents)) & (i1, v1) : sa..(old contents) &
7343     (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 & i1 <
7344     sa..csize) | (sa..csize > i1 & i1 > i2 & (ALL v. ((i1, v) : sa..(old
7345     contents)) = ((i1, v) : sa..contents)) & (i1, v1) : sa..(old contents) &
7346     (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 & i1 <
7347     sa..csize))" */
7348
7349     /*: assume "0 <= i2 & i2 < sb..csize" */
7350     Object r2b = sb.remove_at(i2);
7351     /*: assume "0 <= i1 & i1 < sb..csize" */
7352     sb.set(i1, v1);

```

```

7333     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
7334         */
7335 }
7336
7337 static void set_remove_at_pre_s_204(ArrayList sa, ArrayList sb, int i1, Object v1,
7338     int i2)
7339 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7340     sa..contents = sb..contents & sa..csize = sb..csize"
7341     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7342     ensures "True" */
7343 {
7344     /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, v1) :
7345     sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
7346     > i2 & (ALL v. ((i1, v) : sa..contents) = ((i1 + 1, v) : sa..contents)) &
7347     (i1, v1) : sa..contents & (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 <
7348     sa..csize & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
7349     /*: assume "0 <= i1 & i1 < sa..csize" */
7350     sa.set(i1, v1);
7351     /*: assume "0 <= i2 & i2 < sa..csize" */
7352     sa.remove_at(i2);
7353
7354     sb.remove_at(i2);
7355     sb.set(i1, v1);
7356
7357     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
7358 }
7359
7360 static void set_remove_at_pre_c_204(ArrayList sa, ArrayList sb, int i1, Object v1,
7361     int i2)
7362 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7363     sa..contents = sb..contents & sa..csize = sb..csize"
7364     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7365     ensures "True" */
7366 {
7367     /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, v1) :
7368     sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
7369     > i2 & (ALL v. ((i1, v) : sa..contents) = ((i1 + 1, v) : sa..contents)) &
7370     (i1, v1) : sa..contents & (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 <
7371     sa..csize & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
7372     /*: assume "0 <= i1 & i1 < sa..csize" */
7373     sa.set(i1, v1);
7374     /*: assume "0 <= i2 & i2 < sa..csize" */
7375     sa.remove_at(i2);
7376
7377     /*: assume "0 <= i2 & i2 < sb..csize" */
7378     sb.remove_at(i2);
7379     /*: assume "0 <= i1 & i1 < sb..csize" */
7380     sb.set(i1, v1);
7381
7382     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
7383 }
7384
7385 static void set_remove_at_between_s_205(ArrayList sa, ArrayList sb, int i1, Object
7386     v1, int i2)
7387 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7388     sa..contents = sb..contents & sa..csize = sb..csize"
7389     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7390     ensures "True" */
7391 {
7392     /*: assume "0 <= i1 & i1 < sa..csize" */
7393     sa.set(i1, v1);
7394     /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, v1) :
7395     sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
7396     > i2 & (ALL v. ((i1, v) : sa..(old contents)) = ((i1 + 1, v) :

```

```

7384         sa..contents)) & (i1, v1) : sa..(old contents) & (i1 + 1, v1) : sa..contents
7385         & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
7386     /*: assume "0 <= i2 & i2 < sa..csize" */
7387     sa.remove_at(i2);
7388
7389     sb.remove_at(i2);
7390     sb.set(i1, v1);
7391
7392     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
7393 }
7394
7395 static void set_remove_at_between_c_205(ArrayList sa, ArrayList sb, int i1, Object
7396 v1, int i2)
7397 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7398 sa..contents = sb..contents & sa..csize = sb..csize"
7399 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7400 ensures "True" */
7401 {
7402     /*: assume "0 <= i1 & i1 < sa..csize" */
7403     sa.set(i1, v1);
7404     /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, v1) :
7405 sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
7406 > i2 & (ALL v. ((i1, v) : sa..(old contents)) = ((i1 + 1, v) :
7407 sa..contents)) & (i1, v1) : sa..(old contents) & (i1 + 1, v1) : sa..contents
7408 & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
7409     /*: assume "0 <= i2 & i2 < sa..csize" */
7410     sa.remove_at(i2);
7411
7412     /*: assume "0 <= i2 & i2 < sb..csize" */
7413     sb.remove_at(i2);
7414     /*: assume "0 <= i1 & i1 < sb..csize" */
7415     sb.set(i1, v1);
7416
7417     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
7418 }
7419
7420 static void set_remove_at_post_s_206(ArrayList sa, ArrayList sb, int i1, Object v1,
7421 int i2)
7422 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7423 sa..contents = sb..contents & sa..csize = sb..csize"
7424 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7425 ensures "True" */
7426 {
7427     /*: assume "0 <= i1 & i1 < sa..csize" */
7428     sa.set(i1, v1);
7429     /*: assume "0 <= i2 & i2 < sa..csize" */
7430     sa.remove_at(i2);
7431     /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize & (i1, v1) : sa..contents & 0
7432 <= i1 & i1 < sa..csize) | (sa..csize > i1 & i1 > i2 & (ALL v. ((i1, v) :
7433 sa..(old contents)) = ((i1, v) : sa..(contents))) & (i1, v1) : sa..(old
7434 contents) & (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) & 0 <=
7435 i1 & i1 < sa..csize)" */
7436
7437     sb.remove_at(i2);
7438     sb.set(i1, v1);
7439
7440     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
7441 }
7442
7443 static void set_remove_at_post_c_206(ArrayList sa, ArrayList sb, int i1, Object v1,
7444 int i2)
7445 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7446 sa..contents = sb..contents & sa..csize = sb..csize"
7447 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7448 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 &

```

```

7491         sa..contents = sb..contents & sa..csize = sb..csize"
7492     modifies "sa..contents", "sb..contents"
7493     ensures "True" */
7494 {
7495     /*: assume "0 <= i1 & i1 < sa..csize" */
7496     sa.set(i1, v1);
7497     /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & (i1, v2) :
7498         sa..(old contents) & v1 = v2 & 0 <= i1 & i1 < sa..(old csize)) | i1 > i2" */
7499     /*: assume "0 <= i2 & i2 < sa..csize" */
7500     Object r2a = sa.set(i2, v2);
7501
7502     Object r2b = sb.set(i2, v2);
7503     sb.set(i1, v1);
7504
7505     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
7506 }
7507
7508 static void set_set_between_c_208(ArrayList sa, ArrayList sb, int i1, Object v1,
7509     int i2, Object v2)
7510 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7511     sa..contents = sb..contents & sa..csize = sb..csize"
7512     modifies "sa..contents", "sb..contents"
7513     ensures "True" */
7514 {
7515     /*: assume "0 <= i1 & i1 < sa..csize" */
7516     sa.set(i1, v1);
7517     /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & (i1, v2) :
7518         sa..(old contents) & v1 = v2 & 0 <= i1 & i1 < sa..(old csize)) | i1 > i2)" */
7519     /*: assume "0 <= i2 & i2 < sa..csize" */
7520     Object r2a = sa.set(i2, v2);
7521
7522     /*: assume "0 <= i2 & i2 < sb..csize" */
7523     Object r2b = sb.set(i2, v2);
7524     /*: assume "0 <= i1 & i1 < sb..csize" */
7525     sb.set(i1, v1);
7526
7527     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
7528         */
7529 }
7530
7531 static void set_set_post_s_209(ArrayList sa, ArrayList sb, int i1, Object v1, int
7532     i2, Object v2)
7533 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7534     sa..contents = sb..contents & sa..csize = sb..csize"
7535     modifies "sa..contents", "sb..contents"
7536     ensures "True" */
7537 {
7538     /*: assume "0 <= i1 & i1 < sa..csize" */
7539     sa.set(i1, v1);
7540     /*: assume "0 <= i2 & i2 < sa..csize" */
7541     Object r2a = sa.set(i2, v2);
7542     /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & (i1, v2) :
7543         sa..(old contents) & v1 = v2 & 0 <= i1 & i1 < sa..(old csize)) | i1 > i2" */
7544
7545     Object r2b = sb.set(i2, v2);
7546     sb.set(i1, v1);
7547
7548     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
7549 }
7550
7551 static void set_set_post_c_209(ArrayList sa, ArrayList sb, int i1, Object v1, int
7552     i2, Object v2)
7553 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7554     sa..contents = sb..contents & sa..csize = sb..csize"
7555     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) :
7556         sa..(old contents) & v1 = v2 & 0 <= i1 & i1 < sa..(old csize)) | i1 > i2)" */
7557
7558     /*: assume "0 <= i2 & i2 < sb..csize" */
7559     Object r2b = sb.set(i2, v2);
7560     /*: assume "0 <= i1 & i1 < sb..csize" */
7561     sb.set(i1, v1);
7562
7563     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
7564         */
7565 }
7566
7567 static void set_set_pre_s_210(ArrayList sa, ArrayList sb, int i1, Object v1, int
7568     i2, Object v2)
7569 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7570     sa..contents = sb..contents & sa..csize = sb..csize"
7571     modifies "sa..contents", "sb..contents"
7572     ensures "True" */
7573 {
7574     /*: assume "i1 < i2 | (i1 = i2 & v1 = v2) | i1 > i2" */
7575     /*: assume "0 <= i1 & i1 < sa..csize" */
7576     sa.set(i1, v1);
7577     /*: assume "0 <= i2 & i2 < sa..csize" */
7578     sa.set(i2, v2);
7579
7580     sb.set(i2, v2);
7581     sb.set(i1, v1);
7582
7583     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
7584 }
7585
7586 static void set_set_pre_c_210(ArrayList sa, ArrayList sb, int i1, Object v1, int
7587     i2, Object v2)
7588 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7589     sa..contents = sb..contents & sa..csize = sb..csize"
7590     modifies "sa..contents", "sb..contents"
7591     ensures "True" */
7592 {
7593     /*: assume "~(i1 < i2 | (i1 = i2 & v1 = v2) | i1 > i2)" */
7594     /*: assume "0 <= i1 & i1 < sa..csize" */
7595     sa.set(i1, v1);
7596     /*: assume "0 <= i2 & i2 < sa..csize" */
7597     sa.set(i2, v2);
7598
7599     /*: assume "0 <= i2 & i2 < sb..csize" */
7600     sb.set(i2, v2);
7601     /*: assume "0 <= i1 & i1 < sb..csize" */
7602     sb.set(i1, v1);
7603
7604     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
7605 }
7606
7607 static void set_set_between_s_211(ArrayList sa, ArrayList sb, int i1, Object v1,
7608     int i2, Object v2)
7609 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7610     sa..contents = sb..contents & sa..csize = sb..csize"
7611     modifies "sa..contents", "sb..contents"
7612     ensures "True" */
7613 {

```

```

7609     /*: assume "0 <= i1 & i1 < sa..csize" */
7610     sa.set(i1, v1);
7611     /*: assume "i1 < i2 | (i1 = i2 & v1 = v2) | i1 > i2" */
7612     /*: assume "0 <= i2 & i2 < sa..csize" */
7613     sa.set(i2, v2);
7614
7615     sb.set(i2, v2);
7616     sb.set(i1, v1);
7617
7618     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
7619 }
7620
7621 static void set_set_between_c_211(ArrayList sa, ArrayList sb, int i1, Object v1,
7622     int i2, Object v2)
7623 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7624     sa..contents = sb..contents & sa..csize = sb..csize"
7625     modifies "sa..contents", "sb..contents"
7626     ensures "True" */
7627 {
7628     /*: assume "0 <= i1 & i1 < sa..csize" */
7629     sa.set(i1, v1);
7630     /*: assume "~(i1 < i2 | (i1 = i2 & v1 = v2) | i1 > i2)" */
7631     /*: assume "0 <= i2 & i2 < sa..csize" */
7632     sa.set(i2, v2);
7633
7634     /*: assume "0 <= i2 & i2 < sb..csize" */
7635     sb.set(i2, v2);
7636     /*: assume "0 <= i1 & i1 < sb..csize" */
7637     sb.set(i1, v1);
7638
7639     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
7640 }
7641
7642 static void set_set_post_s_212(ArrayList sa, ArrayList sb, int i1, Object v1, int
7643     i2, Object v2)
7644 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7645     sa..contents = sb..contents & sa..csize = sb..csize"
7646     modifies "sa..contents", "sb..contents"
7647     ensures "True" */
7648 {
7649     /*: assume "0 <= i1 & i1 < sa..csize" */
7650     sa.set(i1, v1);
7651     /*: assume "0 <= i2 & i2 < sa..csize" */
7652     sa.set(i2, v2);
7653     /*: assume "i1 < i2 | (i1 = i2 & v1 = v2) | i1 > i2" */
7654
7655     sb.set(i2, v2);
7656     sb.set(i1, v1);
7657
7658     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
7659 }
7660
7661 static void set_set_post_c_212(ArrayList sa, ArrayList sb, int i1, Object v1, int
7662     i2, Object v2)
7663 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7664     sa..contents = sb..contents & sa..csize = sb..csize"
7665     modifies "sa..contents", "sb..contents"
7666     ensures "True" */
7667 {
7668     /*: assume "0 <= i1 & i1 < sa..csize" */
7669     sa.set(i1, v1);
7670     /*: assume "0 <= i2 & i2 < sa..csize" */
7671     sa.set(i2, v2);
7672     /*: assume "~(i1 < i2 | (i1 = i2 & v1 = v2) | i1 > i2)" */

```

```

7671     /*: assume "0 <= i2 & i2 < sb..csize" */
7672     sb.set(i2, v2);
7673     /*: assume "0 <= i1 & i1 < sb..csize" */
7674     sb.set(i1, v1);
7675
7676     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
7677 }
7678
7679 static void set_size_pre_s_213(ArrayList sa, ArrayList sb, int i1, Object v1)
7680 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7681             sa..contents = sb..contents & sa..csize = sb..csize"
7682 modifies "sa..contents", "sb..contents"
7683 ensures "True" */
7684 {
7685     /*: assume "True" */
7686     /*: assume "0 <= i1 & i1 < sa..csize" */
7687     sa.set(i1, v1);
7688     int r2a = sa.size();
7689
7690     int r2b = sb.size();
7691     sb.set(i1, v1);
7692
7693     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
7694 }
7695
7696 static void set_size_pre_c_213(ArrayList sa, ArrayList sb, int i1, Object v1)
7697 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7698             sa..contents = sb..contents & sa..csize = sb..csize"
7699 modifies "sa..contents", "sb..contents"
7700 ensures "True" */
7701 {
7702     /*: assume "~(True)" */
7703     /*: assume "0 <= i1 & i1 < sa..csize" */
7704     sa.set(i1, v1);
7705     int r2a = sa.size();
7706
7707     int r2b = sb.size();
7708     /*: assume "0 <= i1 & i1 < sb..csize" */
7709     sb.set(i1, v1);
7710
7711     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
7712             */
7713 }
7714
7715 static void set_size_between_s_214(ArrayList sa, ArrayList sb, int i1, Object v1)
7716 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7717             sa..contents = sb..contents & sa..csize = sb..csize"
7718 modifies "sa..contents", "sb..contents"
7719 ensures "True" */
7720 {
7721     /*: assume "0 <= i1 & i1 < sa..csize" */
7722     sa.set(i1, v1);
7723     /*: assume "True" */
7724     int r2a = sa.size();
7725
7726     int r2b = sb.size();
7727     sb.set(i1, v1);
7728
7729     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
7730 }
7731
7732 static void set_size_between_c_214(ArrayList sa, ArrayList sb, int i1, Object v1)
7733 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7734             sa..contents = sb..contents & sa..csize = sb..csize"
7735 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     /*: assert "ria = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
7798 }
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",
7805     "sb..msize"
7806     ensures "True" */
7807 {
7808     /*: assume "~(False)" */
7809     int r1a = sa.size();
7810     /*: assume "0 <= i2 & i2 <= sa..csize" */
7811     sa.add_at(i2, v2);
7812
7813     /*: assume "0 <= i2 & i2 <= sb..csize" */
7814     sb.add_at(i2, v2);
7815     int r1b = sb.size();
7816
7817     /*: assert "~(ria = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
7818     */
7819 }
7820
7821 static void size_add_at_between_s_217(ArrayList sa, ArrayList sb, int i2, Object v2)
7822 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7823     sa..contents = sb..contents & sa..csize = sb..csize"
7824     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
7825     "sb..msize"
7826     ensures "True" */
7827 {
7828     int r1a = sa.size();
7829     /*: assume "False" */
7830     /*: assume "0 <= i2 & i2 <= sa..csize" */
7831     sa.add_at(i2, v2);
7832
7833     sb.add_at(i2, v2);
7834     int r1b = sb.size();
7835
7836     /*: assert "ria = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
7837 }
7838
7839 static void size_add_at_between_c_217(ArrayList sa, ArrayList sb, int i2, Object v2)
7840 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7841     sa..contents = sb..contents & sa..csize = sb..csize"
7842     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
7843     "sb..msize"
7844     ensures "True" */
7845 {
7846     int r1a = sa.size();
7847     /*: assume "~(False)" */
7848     /*: assume "0 <= i2 & i2 <= sa..csize" */
7849     sa.add_at(i2, v2);
7850
7851     /*: assume "0 <= i2 & i2 <= sb..csize" */
7852     sb.add_at(i2, v2);
7853     int r1b = sb.size();
7854
7855     /*: assert "~(ria = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
7856     */
7857 }
7858
7859 static void size_add_at_post_s_218(ArrayList sa, ArrayList sb, int i2, Object v2)
7860 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7861     sa..contents = sb..contents & sa..csize = sb..csize"

```

```

7857     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
7858           "sb..msize"
7859     ensures "True" */
7860 {
7861     int r1a = sa.size();
7862     /*: assume "0 <= i2 & i2 <= sa..csize" */
7863     sa.add_at(i2, v2);
7864     /*: assume "False" */
7865
7866     sb.add_at(i2, v2);
7867     int r1b = sb.size();
7868
7869     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
7870 }
7871
7872 static void size_add_at_post_c_218(ArrayList sa, ArrayList sb, int i2, Object v2)
7873 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7874           sa..contents = sb..contents & sa..csize = sb..csize"
7875     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
7876           "sb..msize"
7877     ensures "True" */
7878 {
7879     int r1a = sa.size();
7880     /*: assume "0 <= i2 & i2 <= sa..csize" */
7881     sa.add_at(i2, v2);
7882     /*: assume "~(False)" */
7883
7884     /*: assume "0 <= i2 & i2 <= sb..csize" */
7885     sb.add_at(i2, v2);
7886     int r1b = sb.size();
7887
7888     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
7889           */
7890 }
7891
7892 static void size_get_pre_s_219(ArrayList sa, ArrayList sb, int i2)
7893 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7894           sa..contents = sb..contents & sa..csize = sb..csize"
7895     ensures "True" */
7896 {
7897     /*: assume "True" */
7898     int r1a = sa.size();
7899     /*: assume "0 <= i2 & i2 < sa..csize" */
7900     Object r2a = sa.get(i2);
7901
7902     Object r2b = sb.get(i2);
7903     int r1b = sb.size();
7904
7905     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
7906           sb..csize" */
7907 }
7908
7909 static void size_get_pre_c_219(ArrayList sa, ArrayList sb, int i2)
7910 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7911           sa..contents = sb..contents & sa..csize = sb..csize"
7912     ensures "True" */
7913 {
7914     /*: assume "~(True)" */
7915     int r1a = sa.size();
7916     /*: assume "0 <= i2 & i2 < sa..csize" */
7917     Object r2a = sa.get(i2);
7918
7919     /*: assume "0 <= i2 & i2 < sb..csize" */
7920     Object r2b = sb.get(i2);
7921     int r1b = sb.size();

```

```

7918     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
7919         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 =
7936         sb..csize" */
7937 }
7938
7939 static void size_get_between_c_220(ArrayList sa, ArrayList sb, int i2)
7940 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7941     sa..contents = sb..contents & sa..csize = sb..csize"
7942     ensures "True" */
7943 {
7944     int r1a = sa.size();
7945     /*: assume "~(True)" */
7946     /*: assume "0 <= i2 & i2 < sa..csize" */
7947     Object r2a = sa.get(i2);
7948
7949     /*: assume "0 <= i2 & i2 < sb..csize" */
7950     Object r2b = sb.get(i2);
7951     int r1b = sb.size();
7952
7953     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
7954         sb..csize)" */
7955 }
7956
7957 static void size_get_post_s_221(ArrayList sa, ArrayList sb, int i2)
7958 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7959     sa..contents = sb..contents & sa..csize = sb..csize"
7960     ensures "True" */
7961 {
7962     int r1a = sa.size();
7963     /*: assume "0 <= i2 & i2 < sa..csize" */
7964     Object r2a = sa.get(i2);
7965     /*: assume "True" */
7966
7967     Object r2b = sb.get(i2);
7968     int r1b = sb.size();
7969
7970     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
7971         sb..csize" */
7972 }
7973
7974 static void size_get_post_c_221(ArrayList sa, ArrayList sb, int i2)
7975 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7976     sa..contents = sb..contents & sa..csize = sb..csize"
7977     ensures "True" */
7978 {
7979     int r1a = sa.size();
7980     /*: assume "0 <= i2 & i2 < sa..csize" */
7981     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 =
7986         sb..csize)" */
7987 }
7988
7989 static void size_index0f_pre_s_222(ArrayList sa, ArrayList sb, Object v2)
7990 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7991     sa..contents = sb..contents & sa..csize = sb..csize"
7992     ensures "True" */
7993 {
7994     /*: assume "True" */
7995     int r1a = sa.size();
7996     int r2a = sa.index0f(v2);
7997
7998     int r2b = sb.index0f(v2);
7999     int r1b = sb.size();
8000
8001     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
8002         sb..csize" */
8003 }
8004
8005 static void size_index0f_pre_c_222(ArrayList sa, ArrayList sb, Object v2)
8006 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8007     sa..contents = sb..contents & sa..csize = sb..csize"
8008     ensures "True" */
8009 {
8010     /*: assume "~(True)" */
8011     int r1a = sa.size();
8012     int r2a = sa.index0f(v2);
8013
8014     int r2b = sb.index0f(v2);
8015     int r1b = sb.size();
8016
8017     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
8018         sb..csize)" */
8019 }
8020
8021 static void size_index0f_between_s_223(ArrayList sa, ArrayList sb, Object v2)
8022 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8023     sa..contents = sb..contents & sa..csize = sb..csize"
8024     ensures "True" */
8025 {
8026     int r1a = sa.size();
8027     /*: assume "True" */
8028     int r2a = sa.index0f(v2);
8029
8030     int r2b = sb.index0f(v2);
8031     int r1b = sb.size();
8032
8033     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
8034         sb..csize" */
8035 }
8036
8037 static void size_index0f_between_c_223(ArrayList sa, ArrayList sb, Object v2)
8038 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8039     sa..contents = sb..contents & sa..csize = sb..csize"
8040     ensures "True" */
8041 {
8042     int r1a = sa.size();
8043     /*: 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     int r2b = sb.lastIndexOf(v2);
8102     int r1b = sb.size();
8103
8104     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
8105         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 =
8121         sb..csize" */
8122 }
8123
8124 static void size_lastIndexOf_between_c_226(ArrayList sa, ArrayList sb, Object v2)
8125 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8126     sa..contents = sb..contents & sa..csize = sb..csize"
8127     ensures "True" */
8128 {
8129     int r1a = sa.size();
8130     /*: assume "~(True)" */
8131     int r2a = sa.lastIndexOf(v2);
8132
8133     int r2b = sb.lastIndexOf(v2);
8134     int r1b = sb.size();
8135
8136     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
8137         sb..csize)" */
8138 }
8139
8140 static void size_lastIndexOf_post_s_227(ArrayList sa, ArrayList sb, Object v2)
8141 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8142     sa..contents = sb..contents & sa..csize = sb..csize"
8143     ensures "True" */
8144 {
8145     int r1a = sa.size();
8146     int r2a = sa.lastIndexOf(v2);
8147     /*: assume "True" */
8148
8149     int r2b = sb.lastIndexOf(v2);
8150     int r1b = sb.size();
8151
8152     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
8153         sb..csize" */
8154 }
8155
8156 static void size_lastIndexOf_post_c_227(ArrayList sa, ArrayList sb, Object v2)
8157 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8158     sa..contents = sb..contents & sa..csize = sb..csize"
8159     ensures "True" */
8160 {
8161     int r1a = sa.size();
8162     int r2a = sa.lastIndexOf(v2);
8163     /*: assume "~(True)" */
8164 }

```

```

8162     int r2b = sb.lastIndexOf(v2);
8163     int r1b = sb.size();
8164
8165     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
      sb..csize)" */
8166 }
8167
8168 static void size_remove_at_pre_s_228(ArrayList sa, ArrayList sb, int i2)
8169 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8170     sa..contents = sb..contents & sa..csize = sb..csize"
8171     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
8172     ensures "True" */
8173 {
8174     /*: assume "False" */
8175     int r1a = sa.size();
8176     /*: assume "0 <= i2 & i2 < sa..csize" */
8177     Object r2a = sa.remove_at(i2);
8178
8179     Object r2b = sb.remove_at(i2);
8180     int r1b = sb.size();
8181
8182     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
      sb..csize" */
8183 }
8184
8185 static void size_remove_at_pre_c_228(ArrayList sa, ArrayList sb, int i2)
8186 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8187     sa..contents = sb..contents & sa..csize = sb..csize"
8188     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
8189     ensures "True" */
8190 {
8191     /*: assume "~(False)" */
8192     int r1a = sa.size();
8193     /*: assume "0 <= i2 & i2 < sa..csize" */
8194     Object r2a = sa.remove_at(i2);
8195
8196     /*: assume "0 <= i2 & i2 < sb..csize" */
8197     Object r2b = sb.remove_at(i2);
8198     int r1b = sb.size();
8199
8200     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
      sb..csize)" */
8201 }
8202
8203 static void size_remove_at_between_s_229(ArrayList sa, ArrayList sb, int i2)
8204 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8205     sa..contents = sb..contents & sa..csize = sb..csize"
8206     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
8207     ensures "True" */
8208 {
8209     int r1a = sa.size();
8210     /*: assume "False" */
8211     /*: assume "0 <= i2 & i2 < sa..csize" */
8212     Object r2a = sa.remove_at(i2);
8213
8214     Object r2b = sb.remove_at(i2);
8215     int r1b = sb.size();
8216
8217     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
      sb..csize" */
8218 }
8219
8220 static void size_remove_at_between_c_229(ArrayList sa, ArrayList sb, int i2)
8221 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8222     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 =
8236         sb..csize)" */
8237 }
8238
8239 static void size_remove_at_post_s_230(ArrayList sa, ArrayList sb, int i2)
8240 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8241     sa..contents = sb..contents & sa..csize = sb..csize"
8242     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
8243     ensures "True" */
8244 {
8245     int r1a = sa.size();
8246     /*: assume "0 <= i2 & i2 < sa..csize" */
8247     Object r2a = sa.remove_at(i2);
8248     /*: assume "False" */
8249
8250     Object r2b = sb.remove_at(i2);
8251     int r1b = sb.size();
8252
8253     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
8254         sb..csize" */
8255 }
8256
8257 static void size_remove_at_post_c_230(ArrayList sa, ArrayList sb, int i2)
8258 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8259     sa..contents = sb..contents & sa..csize = sb..csize"
8260     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
8261     ensures "True" */
8262 {
8263     int r1a = sa.size();
8264     /*: assume "0 <= i2 & i2 < sa..csize" */
8265     Object r2a = sa.remove_at(i2);
8266     /*: assume "~(False)" */
8267
8268     /*: assume "0 <= i2 & i2 < sb..csize" */
8269     Object r2b = sb.remove_at(i2);
8270     int r1b = sb.size();
8271
8272     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
8273         sb..csize)" */
8274 }
8275
8276 static void size_remove_at_pre_s_231(ArrayList sa, ArrayList sb, int i2)
8277 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8278     sa..contents = sb..contents & sa..csize = sb..csize"
8279     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
8280     ensures "True" */
8281 {
8282     /*: assume "False" */
8283     int r1a = sa.size();
8284     /*: assume "0 <= i2 & i2 < sa..csize" */
8285     sa.remove_at(i2);
8286
8287     sb.remove_at(i2);

```

```

8285     int r1b = sb.size();
8286
8287     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
8288 }
8289
8290 static void size_remove_at_pre_c_231(ArrayList sa, ArrayList sb, int i2)
8291 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8292     sa..contents = sb..contents & sa..csize = sb..csize"
8293     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
8294     ensures "True" */
8295 {
8296     /*: assume "~(False)" */
8297     int r1a = sa.size();
8298     /*: assume "0 <= i2 & i2 < sa..csize" */
8299     sa.remove_at(i2);
8300
8301     /*: assume "0 <= i2 & i2 < sb..csize" */
8302     sb.remove_at(i2);
8303     int r1b = sb.size();
8304
8305     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
8306         */
8307 }
8308
8309 static void size_remove_at_between_s_232(ArrayList sa, ArrayList sb, int i2)
8310 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8311     sa..contents = sb..contents & sa..csize = sb..csize"
8312     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
8313     ensures "True" */
8314 {
8315     int r1a = sa.size();
8316     /*: assume "False" */
8317     /*: assume "0 <= i2 & i2 < sa..csize" */
8318     sa.remove_at(i2);
8319
8320     sb.remove_at(i2);
8321     int r1b = sb.size();
8322
8323     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
8324 }
8325
8326 static void size_remove_at_between_c_232(ArrayList sa, ArrayList sb, int i2)
8327 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8328     sa..contents = sb..contents & sa..csize = sb..csize"
8329     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
8330     ensures "True" */
8331 {
8332     int r1a = sa.size();
8333     /*: assume "~(False)" */
8334     /*: assume "0 <= i2 & i2 < sa..csize" */
8335     sa.remove_at(i2);
8336
8337     /*: assume "0 <= i2 & i2 < sb..csize" */
8338     sb.remove_at(i2);
8339     int r1b = sb.size();
8340
8341     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
8342         */
8343 }
8344
8345 static void size_remove_at_post_s_233(ArrayList sa, ArrayList sb, int i2)
8346 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8347     sa..contents = sb..contents & sa..csize = sb..csize"
8348     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
8349     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
8379 static void size_set_pre_s_234(ArrayList sa, ArrayList sb, int i2, Object v2)
8380 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8381     sa..contents = sb..contents & sa..csize = sb..csize"
8382     modifies "sa..contents", "sb..contents"
8383     ensures "True" */
8384 {
8385     /*: assume "True" */
8386     int r1a = sa.size();
8387     /*: assume "0 <= i2 & i2 < sa..csize" */
8388     Object r2a = sa.set(i2, v2);
8389
8390     Object r2b = sb.set(i2, v2);
8391     int r1b = sb.size();
8392
8393     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
8394         sb..csize" */
8395 }
8396
8397 static void size_set_pre_c_234(ArrayList sa, ArrayList sb, int i2, Object v2)
8398 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8399     sa..contents = sb..contents & sa..csize = sb..csize"
8400     modifies "sa..contents", "sb..contents"
8401     ensures "True" */
8402 {
8403     /*: assume "~(True)" */
8404     int r1a = sa.size();
8405     /*: assume "0 <= i2 & i2 < sa..csize" */
8406     Object r2a = sa.set(i2, v2);
8407
8408     /*: assume "0 <= i2 & i2 < sb..csize" */
8409     Object r2b = sb.set(i2, v2);
8410     int r1b = sb.size();

```

```

8410     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
8411         sb..csize)" */
8412 }
8413 static void size_set_between_s_235(ArrayList sa, ArrayList sb, int i2, Object v2)
8414 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8415         sa..contents = sb..contents & sa..csize = sb..csize"
8416     modifies "sa..contents", "sb..contents"
8417     ensures "True" */
8418 {
8419     int r1a = sa.size();
8420     /*: assume "True" */
8421     /*: assume "0 <= i2 & i2 < sa..csize" */
8422     Object r2a = sa.set(i2, v2);
8423
8424     Object r2b = sb.set(i2, v2);
8425     int r1b = sb.size();
8426
8427     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
8428         sb..csize" */
8429 }
8430 static void size_set_between_c_235(ArrayList sa, ArrayList sb, int i2, Object v2)
8431 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8432         sa..contents = sb..contents & sa..csize = sb..csize"
8433     modifies "sa..contents", "sb..contents"
8434     ensures "True" */
8435 {
8436     int r1a = sa.size();
8437     /*: assume "~(True)" */
8438     /*: assume "0 <= i2 & i2 < sa..csize" */
8439     Object r2a = sa.set(i2, v2);
8440
8441     /*: assume "0 <= i2 & i2 < sb..csize" */
8442     Object r2b = sb.set(i2, v2);
8443     int r1b = sb.size();
8444
8445     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
8446         sb..csize)" */
8447 }
8448 static void size_set_post_s_236(ArrayList sa, ArrayList sb, int i2, Object v2)
8449 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8450         sa..contents = sb..contents & sa..csize = sb..csize"
8451     modifies "sa..contents", "sb..contents"
8452     ensures "True" */
8453 {
8454     int r1a = sa.size();
8455     /*: assume "0 <= i2 & i2 < sa..csize" */
8456     Object r2a = sa.set(i2, v2);
8457     /*: assume "True" */
8458
8459     Object r2b = sb.set(i2, v2);
8460     int r1b = sb.size();
8461
8462     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
8463         sb..csize" */
8464 }
8465 static void size_set_post_c_236(ArrayList sa, ArrayList sb, int i2, Object v2)
8466 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8467         sa..contents = sb..contents & sa..csize = sb..csize"
8468     modifies "sa..contents", "sb..contents"
8469     ensures "True" */
8470 {

```

```

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
8554 static void size_set_post_s_239(ArrayList sa, ArrayList sb, int i2, Object v2)
8555 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8556      sa..contents = sb..contents & sa..csize = sb..csize"
8557      modifies "sa..contents", "sb..contents"
8558      ensures "True" */
8559 {
8560     int r1a = sa.size();
8561     /*: assume "0 <= i2 & i2 < sa..csize" */
8562     sa.set(i2, v2);
8563     /*: assume "True" */
8564
8565     sb.set(i2, v2);
8566     int r1b = sb.size();
8567
8568     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
8569 }
8570
8571 static void size_set_post_c_239(ArrayList sa, ArrayList sb, int i2, Object v2)
8572 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8573      sa..contents = sb..contents & sa..csize = sb..csize"
8574      modifies "sa..contents", "sb..contents"
8575      ensures "True" */
8576 {
8577     int r1a = sa.size();
8578     /*: assume "0 <= i2 & i2 < sa..csize" */
8579     sa.set(i2, v2);
8580     /*: assume "~(True)" */
8581
8582     /*: assume "0 <= i2 & i2 < sb..csize" */
8583     sb.set(i2, v2);
8584     int r1b = sb.size();
8585
8586     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
8587      */
8588 }
8589
8590 static void size_size_pre_s_240(ArrayList sa, ArrayList sb)
8591 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8592      sa..contents = sb..contents & sa..csize = sb..csize"
8593      ensures "True" */
8594 {
8595     /*: assume "True" */
8596     int r1a = sa.size();
8597     int r2a = sa.size();

```

```

8597     int r2b = sb.size();
8598     int r1b = sb.size();
8599
8600     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
8601         sb..csize" */
8602 }
8603
8604 static void size_size_pre_c_240(ArrayList sa, ArrayList sb)
8605 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8606     sa..contents = sb..contents & sa..csize = sb..csize"
8607     ensures "True" */
8608 {
8609     /*: assume "~(True)" */
8610     int r1a = sa.size();
8611     int r2a = sa.size();
8612
8613     int r2b = sb.size();
8614     int r1b = sb.size();
8615
8616     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
8617         sb..csize)" */
8618 }
8619
8620 static void size_size_between_s_241(ArrayList sa, ArrayList sb)
8621 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8622     sa..contents = sb..contents & sa..csize = sb..csize"
8623     ensures "True" */
8624 {
8625     int r1a = sa.size();
8626     /*: assume "True" */
8627     int r2a = sa.size();
8628
8629     int r2b = sb.size();
8630     int r1b = sb.size();
8631
8632     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
8633         sb..csize" */
8634 }
8635
8636 static void size_size_between_c_241(ArrayList sa, ArrayList sb)
8637 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8638     sa..contents = sb..contents & sa..csize = sb..csize"
8639     ensures "True" */
8640 {
8641     int r1a = sa.size();
8642     /*: assume "~(True)" */
8643     int r2a = sa.size();
8644
8645     int r2b = sb.size();
8646     int r1b = sb.size();
8647
8648     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
8649         sb..csize)" */
8650 }
8651
8652 static void size_size_post_s_242(ArrayList sa, ArrayList sb)
8653 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
8654     sa..contents = sb..contents & sa..csize = sb..csize"
8655     ensures "True" */
8656 {
8657     int r1a = sa.size();
8658     int r2a = sa.size();
8659     /*: assume "True" */
8660
8661     int r2b = sb.size();

```

```

8658     int r1b = sb.size();
8659
8660     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
           sb..csize" */
8661 }
8662
8663 static void size_size_post_c_242(ArrayList sa, ArrayList sb)
8664 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
           sa..contents = sb..contents & sa..csize = sb..csize"
8665     ensures "True" */
8666 {
8667     int r1a = sa.size();
8668     int r2a = sa.size();
8669     /*: assume "~(True)" */
8670
8671     int r2b = sb.size();
8672     int r1b = sb.size();
8673
8674     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
           sb..csize)" */
8675 }
8676
8677 }
8678

```

Listing 18. ArrayListComm.java

```

1 class ArrayListComm {
2     static void add_at_add_at_pre_s_0(ArrayList sa, ArrayList sb, int i1, Object v1,
           int i2, Object v2)
3     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
           sa..contents = sb..contents & sa..csize = sb..csize"
4     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
           "sb..msize"
5     ensures "True" */
6
7     {
8         /*: 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)" */
9         /*: assume "0 <= i1 & i1 <= sa..csize" */
10        sa.add_at(i1, v1);
11        /*: assume "0 <= i2 & i2 <= sa..csize" */
12        sa.add_at(i2, v2);
13
14        sb.add_at(i2, v2);
15        sb.add_at(i1, v1);
16
17        /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
18    }
19
20    static void add_at_add_at_pre_c_0(ArrayList sa, ArrayList sb, int i1, Object v1,
           int i2, Object v2)
21    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
           sa..contents = sb..contents & sa..csize = sb..csize"
22    modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
           "sb..msize"
23    ensures "True" */
24
25    {
26        /*: 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))" */
27        /*: assume "0 <= i1 & i1 <= sa..csize" */
28        sa.add_at(i1, v1);
29        /*: assume "0 <= i2 & i2 <= sa..csize" */
30        sa.add_at(i2, v2);
31

```

```

32     /*: assume "0 <= i2 & i2 <= sb..csize" */
33     sb.add_at(i2, v2);
34     /*: assume "0 <= i1 & i1 <= sb..csize" */
35     sb.add_at(i1, v1);
36
37     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
38 }
39
40 static void add_at_add_at_between_s_1(ArrayList sa, ArrayList sb, int i1, Object
41     v1, int i2, Object v2)
42 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
43     sa..contents = sb..contents & sa..csize = sb..csize"
44     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
45     "sb..msize"
46     ensures "True" */
47 {
48     /*: assume "0 <= i1 & i1 <= sa..csize" */
49     sa.add_at(i1, v1);
50     /*: assume "(i1 < i2 & i2 <= sa..csize - 1 & (i2, v2) : sa..contents & 0 <= i2
51     & i2 < sa..csize) | (i1 = i2 & v1 = v2) | (i1 > i2 & (i1 - 1, v1) :
52     sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize)" */
53     /*: assume "0 <= i2 & i2 <= sa..csize" */
54     sa.add_at(i2, v2);
55
56     sb.add_at(i2, v2);
57     sb.add_at(i1, v1);
58
59     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
60 }
61
62 static void add_at_add_at_between_c_1(ArrayList sa, ArrayList sb, int i1, Object
63     v1, int i2, Object v2)
64 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
65     sa..contents = sb..contents & sa..csize = sb..csize"
66     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
67     "sb..msize"
68     ensures "True" */
69 {
70     /*: assume "0 <= i1 & i1 <= sa..csize" */
71     sa.add_at(i1, v1);
72     /*: assume "~((i1 < i2 & i2 <= sa..csize - 1 & (i2, v2) : sa..contents & 0 <=
73     i2 & i2 < sa..csize) | (i1 = i2 & v1 = v2) | (i1 > i2 & (i1 - 1, v1) :
74     sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize))" */
75     /*: assume "0 <= i2 & i2 <= sa..csize" */
76     sa.add_at(i2, v2);
77
78     /*: assume "0 <= i2 & i2 <= sb..csize" */
79     sb.add_at(i2, v2);
80     /*: assume "0 <= i1 & i1 <= sb..csize" */
81     sb.add_at(i1, v1);
82
83     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
84 }
85
86 static void add_at_add_at_post_s_2(ArrayList sa, ArrayList sb, int i1, Object v1,
87     int i2, Object v2)
88 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
89     sa..contents = sb..contents & sa..csize = sb..csize"
90     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
91     "sb..msize"
92     ensures "True" */
93 {
94     /*: assume "0 <= i1 & i1 <= sa..csize" */
95     sa.add_at(i1, v1);
96     /*: 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,
97     int i2, Object v2)
98 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
99     sa..contents = sb..contents & sa..csize = sb..csize"
100    modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
101     "sb..msize"
102    ensures "True" */
103 {
104     /*: assume "0 <= i1 & i1 <= sa..csize" */
105     sa.add_at(i1, v1);
106     /*: assume "0 <= i2 & i2 <= sa..csize" */
107     sa.add_at(i2, v2);
108     /*: assume "~((i1 < i2 & i2 <= sa..csize - 2 & (i2 + 1, v2) : sa..contents & 0
109     <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & v1 = v2) | (i1 > i2 & (i1, v1)
110     : sa..contents & 0 <= i1 & i1 < sa..csize))" */
111
112     /*: assume "0 <= i2 & i2 <= sb..csize" */
113     sb.add_at(i2, v2);
114     /*: assume "0 <= i1 & i1 <= sb..csize" */
115     sb.add_at(i1, v1);
116
117     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
118 }
119
120 static void add_at_get_pre_s_3(ArrayList sa, ArrayList sb, int i1, Object v1, int
121     i2)
122 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
123     sa..contents = sb..contents & sa..csize = sb..csize"
124    modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
125     "sb..msize"
126    ensures "True" */
127 {
128     /*: assume "(i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents) =
129     ((i2, v) : sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2
130     < sa..csize) | (i1 = i2 & i2 < sa..csize & (i1, v1) : sa..contents & 0 <= i1
131     & i1 < sa..csize) | i1 > i2" */
132     /*: assume "0 <= i1 & i1 <= sa..csize" */
133     sa.add_at(i1, v1);
134     /*: assume "0 <= i2 & i2 < sa..csize" */
135     Object r2a = sa.get(i2);
136
137     Object r2b = sb.get(i2);
138     sb.add_at(i1, v1);
139
140     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
141 }
142
143 static void add_at_get_pre_c_3(ArrayList sa, ArrayList sb, int i1, Object v1, int
144     i2)
145 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
146     sa..contents = sb..contents & sa..csize = sb..csize"
147    modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
148     "sb..msize"
149    ensures "True" */

```

```

139 {
140     /*: 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)" */
141     /*: assume "0 <= i1 & i1 <= sa..csize" */
142     sa.add_at(i1, v1);
143     /*: assume "0 <= i2 & i2 < sa..csize" */
144     Object r2a = sa.get(i2);
145
146     /*: assume "0 <= i2 & i2 < sb..csize" */
147     Object r2b = sb.get(i2);
148     /*: assume "0 <= i1 & i1 <= sb..csize" */
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_get_between_s_4(ArrayList sa, ArrayList sb, int i1, Object v1,
    int i2)
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",
158     "sb..msize"
159     ensures "True" */
160 {
161     /*: assume "0 <= i1 & i1 <= sa..csize" */
162     sa.add_at(i1, v1);
163     /*: 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" */
164     /*: assume "0 <= i2 & i2 < sa..csize" */
165     Object r2a = sa.get(i2);
166
167     Object r2b = sb.get(i2);
168     sb.add_at(i1, v1);
169
170     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
171 }
172
173 static void add_at_get_between_c_4(ArrayList sa, ArrayList sb, int i1, Object v1,
    int i2)
174 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
175     sa..contents = sb..contents & sa..csize = sb..csize"
176     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
177     "sb..msize"
178     ensures "True" */
179 {
180     /*: assume "0 <= i1 & i1 <= sa..csize" */
181     sa.add_at(i1, v1);
182     /*: 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)" */
183     /*: assume "0 <= i2 & i2 < sa..csize" */
184     Object r2a = sa.get(i2);
185
186     /*: assume "0 <= i2 & i2 < sb..csize" */
187     Object r2b = sb.get(i2);
188     /*: assume "0 <= i1 & i1 <= sb..csize" */
189     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
193     i2)
194 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
195     sa..contents = sb..contents & sa..csize = sb..csize"
196     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
197     "sb..msize"
198     ensures "True" */
199 {
200     /*: assume "0 <= i1 & i1 <= sa..csize" */
201     sa.add_at(i1, v1);
202     /*: assume "0 <= i2 & i2 < sa..csize" */
203     Object r2a = sa.get(i2);
204     /*: assume "(i1 < i2 & i2 < sa..csize - 1 & (i2 + 1, r2a) : sa..contents & 0 <=
205         i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, v1)
206         : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | i1 > i2" */
207
208     Object r2b = sb.get(i2);
209     sb.add_at(i1, v1);
210
211     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
212 }
213
214 static void add_at_get_post_c_5(ArrayList sa, ArrayList sb, int i1, Object v1, int
215     i2)
216 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
217     sa..contents = sb..contents & sa..csize = sb..csize"
218     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
219     "sb..msize"
220     ensures "True" */
221 {
222     /*: assume "0 <= i1 & i1 <= sa..csize" */
223     sa.add_at(i1, v1);
224     /*: assume "0 <= i2 & i2 < sa..csize" */
225     Object r2a = sa.get(i2);
226     /*: assume "~((i1 < i2 & i2 < sa..csize - 1 & (i2 + 1, r2a) : sa..contents & 0
227         <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1,
228         v1) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | i1 > i2)" */
229
230     /*: assume "0 <= i2 & i2 < sb..csize" */
231     Object r2b = sb.get(i2);
232     /*: assume "0 <= i1 & i1 <= sb..csize" */
233     sb.add_at(i1, v1);
234
235     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
236         */
237 }
238
239 static void add_at_indexOf_pre_s_6(ArrayList sa, ArrayList sb, int i1, Object v1,
240     Object v2)
241 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
242     sa..contents = sb..contents & sa..csize = sb..csize"
243     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
244     "sb..msize"
245     ensures "True" */
246 {
247     /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
248         v2) | (EX i. (i, v2) : sa..contents & 0 <= i & i < i1) | ~(EX i. (i, v2) :
249         sa..contents & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 <
250         sa..csize & v1 = v2)" */
251     /*: assume "0 <= i1 & i1 <= sa..csize" */
252     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_c_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     /*: assume "0 <= i1 & i1 <= sb..csize" */
278     sb.add_at(i1, v1);
279
280     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
281 }
282
283 static void add_at_indexOf_post_c_8(ArrayList sa, ArrayList sb, int i1, Object v1,
    Object v2)
284 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
285     sa..contents = sb..contents & sa..csize = sb..csize"
286 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
    "sb..msize"
287 ensures "True" */
288 {
289     /*: 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) :
293         sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
294
295     int r2b = sb.indexOf(v2);
296     /*: assume "0 <= i1 & i1 <= sb..csize" */
297     sb.add_at(i1, v1);
298
299     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
300         */
301 }
302
303 static void add_at_lastIndexOf_pre_s_9(ArrayList sa, ArrayList sb, int i1, Object
304     v1, Object v2)
305 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
306     sa..contents = sb..contents & sa..csize = sb..csize"
307     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
308     "sb..msize"
309     ensures "True" */
310 {
311     /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
312         v2 | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
313         sa..contents & i1 <= i & i < sa..csize) & v1 ~= v2)" */
314     /*: assume "0 <= i1 & i1 <= sa..csize" */
315     sa.add_at(i1, v1);
316     int r2a = sa.lastIndexOf(v2);
317
318     int r2b = sb.lastIndexOf(v2);
319     sb.add_at(i1, v1);
320
321     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
322 }
323
324 static void add_at_remove_at_pre_s_12(ArrayList sa, ArrayList sb, int i1, Object
325     v1, int i2)
326 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
327     sa..contents = sb..contents & sa..csize = sb..csize"
328     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
329     "sb..msize"
330     ensures "True" */
331 {
332     /*: assume "(i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents) =
333         ((i2, v) : sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2
334         < sa..csize) | (sa..csize - 1 >= i1 & i1 = i2 & (i1, v1) : sa..contents & 0
335         <= i1 & i1 < sa..csize) | (sa..csize - 1 >= i1 & i1 > i2 & (i1, v1) :
336         sa..contents & 0 <= i1 & i1 < sa..csize)" */
337     /*: assume "0 <= i1 & i1 <= sa..csize" */
338     sa.add_at(i1, v1);
339     /*: assume "0 <= i2 & i2 < sa..csize" */
340     Object r2a = sa.remove_at(i2);
341
342     Object r2b = sb.remove_at(i2);
343     sb.add_at(i1, v1);
344
345     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
346 }
347
348 static void add_at_remove_at_pre_c_12(ArrayList sa, ArrayList sb, int i1, Object
349     v1, int i2)
350 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
351     sa..contents = sb..contents & sa..csize = sb..csize"
352     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
353     "sb..msize"
354     ensures "True" */

```

```

341 {
342     /*: 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))" */
343     /*: assume "0 <= i1 & i1 <= sa..csize" */
344     sa.add_at(i1, v1);
345     /*: assume "0 <= i2 & i2 < sa..csize" */
346     Object r2a = sa.remove_at(i2);
347
348     /*: assume "0 <= i2 & i2 < sb..csize" */
349     Object r2b = sb.remove_at(i2);
350     /*: assume "0 <= i1 & i1 <= sb..csize" */
351     sb.add_at(i1, v1);
352
353     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
354 }
355
356 static void add_at_remove_at_between_s_13(ArrayList sa, ArrayList sb, int i1,
    Object v1, int i2)
357 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
358     sa..contents = sb..contents & sa..csize = sb..csize"
359     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
360     "sb..msize"
361     ensures "True" */
362 {
363     /*: assume "0 <= i1 & i1 <= sa..csize" */
364     sa.add_at(i1, v1);
365     /*: 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))" */
366     /*: assume "0 <= i2 & i2 < sa..csize" */
367     Object r2a = sa.remove_at(i2);
368
369     Object r2b = sb.remove_at(i2);
370     sb.add_at(i1, v1);
371
372     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
373 }
374
375 static void add_at_remove_at_between_c_13(ArrayList sa, ArrayList sb, int i1,
    Object v1, int i2)
376 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
377     sa..contents = sb..contents & sa..csize = sb..csize"
378     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
379     "sb..msize"
380     ensures "True" */
381 {
382     /*: assume "0 <= i1 & i1 <= sa..csize" */
383     sa.add_at(i1, v1);
384     /*: 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))" */
385     /*: assume "0 <= i2 & i2 < sa..csize" */
386     Object r2a = sa.remove_at(i2);
387
388     /*: assume "0 <= i2 & i2 < sb..csize" */
389     Object r2b = sb.remove_at(i2);
390     /*: assume "0 <= i1 & i1 <= sb..csize" */

```

```

389     sb.add_at(i1, v1);
390
391     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
392         */
393 }
394
395 static void add_at_remove_at_post_s_14(ArrayList sa, ArrayList sb, int i1, Object
396     v1, int i2)
397 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
398     sa..contents = sb..contents & sa..csize = sb..csize"
399 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
400     "sb..msize"
401 ensures "True" */
402 {
403     /*: assume "0 <= i1 & i1 <= sa..csize" */
404     sa.add_at(i1, v1);
405     /*: assume "0 <= i2 & i2 < sa..csize" */
406     Object r2a = sa.remove_at(i2);
407     /*: assume "(i1 < i2 & i2 < sa..csize & (i2, r2a) : sa..contents & 0 <= i2 & i2
408         < sa..csize) | (sa..csize - 1 >= i1 & i1 = i2 & (i1, v1) : sa..contents & 0
409         <= i1 & i1 < sa..csize) | (sa..csize - 1 >= i1 & i1 > i2 & (i1, v1) :
410         sa..contents & 0 <= i1 & i1 < sa..csize)" */
411
412     Object r2b = sb.remove_at(i2);
413     sb.add_at(i1, v1);
414
415     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
416 }
417
418 static void add_at_remove_at_post_c_14(ArrayList sa, ArrayList sb, int i1, Object
419     v1, int i2)
420 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
421     sa..contents = sb..contents & sa..csize = sb..csize"
422 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
423     "sb..msize"
424 ensures "True" */
425 {
426     /*: assume "0 <= i1 & i1 <= sa..csize" */
427     sa.add_at(i1, v1);
428     /*: assume "0 <= i2 & i2 < sa..csize" */
429     Object r2a = sa.remove_at(i2);
430     /*: assume "~((i1 < i2 & i2 < sa..csize & (i2, r2a) : sa..contents & 0 <= i2 &
431         i2 < sa..csize) | (sa..csize - 1 >= i1 & i1 = i2 & (i1, v1) : sa..contents &
432         0 <= i1 & i1 < sa..csize) | (sa..csize - 1 >= i1 & i1 > i2 & (i1, v1) :
433         sa..contents & 0 <= i1 & i1 < sa..csize))" */
434
435     /*: assume "0 <= i2 & i2 < sb..csize" */
436     Object r2b = sb.remove_at(i2);
437     /*: assume "0 <= i1 & i1 <= sb..csize" */
438     sb.add_at(i1, v1);
439
440     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
441         */
442 }
443
444 static void add_at_remove_at_pre_s_15(ArrayList sa, ArrayList sb, int i1, Object
445     v1, int i2)
446 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
447     sa..contents = sb..contents & sa..csize = sb..csize"
448 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
449     "sb..msize"
450 ensures "True" */
451 {
452     /*: assume "(i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents) =
453         ((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)" */
439 /*: assume "0 <= i1 & i1 <= sa..csize" */
440 sa.add_at(i1, v1);
441 /*: assume "0 <= i2 & i2 < sa..csize" */
442 sa.remove_at(i2);
443
444 sb.remove_at(i2);
445 sb.add_at(i1, v1);
446
447 /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
448 }
449
450 static void add_at_remove_at_between_s_16(ArrayList sa, ArrayList sb, int i1,
    Object v1, int i2)
451 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
452     sa..contents = sb..contents & sa..csize = sb..csize"
453     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
454     "sb..msize"
455     ensures "True" */
456 {
457     /*: assume "0 <= i1 & i1 <= sa..csize" */
458     sa.add_at(i1, v1);
459     /*: assume "(i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa..contents) =
460         ((i2 + 1, v) : sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2
461         + 1 < sa..csize) | (sa..csize - 2 >= i1 & i1 = i2 & (i1 + 1, v1) :
462         sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 2 >= i1 & i1
463         > i2 & (i1 + 1, v1) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
464     /*: assume "0 <= i2 & i2 < sa..csize" */
465     sa.remove_at(i2);
466
467     sb.remove_at(i2);
468     sb.add_at(i1, v1);
469
470     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
471 }
472
473 static void add_at_remove_at_post_s_17(ArrayList sa, ArrayList sb, int i1, Object
    v1, int i2)
474 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
475     sa..contents = sb..contents & sa..csize = sb..csize"
476     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
477     "sb..msize"
478     ensures "True" */
479 {
480     /*: assume "0 <= i1 & i1 <= sa..csize" */
481     sa.add_at(i1, v1);
482     /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
483     /*: ghost specvar sa__csize :: "int" = "sa..csize" */
484     /*: assume "0 <= i2 & i2 < sa..csize" */
485     sa.remove_at(i2);
486     /*: assume "(i1 < i2 & i2 < sa..csize & (ALL v. ((i2, v) : sa__contents) =
487         ((i2, v) : sa..contents)) & 0 <= i2 & i2 < sa__csize & 0 <= i2 & i2 <
488         sa..csize) | (sa..csize - 1 >= i1 & i1 = i2 & (i1, v1) : sa..contents & 0 <=
489         i1 & i1 < sa..csize) | (sa..csize - 1 >= i1 & i1 > i2 & (i1, v1) :
490         sa..contents & 0 <= i1 & i1 < sa..csize)" */
491
492     sb.remove_at(i2);
493     sb.add_at(i1, v1);
494
495     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
496 }
497

```

```

488 static void add_at_remove_at_post_c_17(ArrayList sa, ArrayList sb, int i1, Object
489     v1, int i2)
490 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
491     sa..contents = sb..contents & sa..csize = sb..csize"
492     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
493     "sb..msize"
494     ensures "True" */
495 {
496     /*: assume "0 <= i1 & i1 <= sa..csize" */
497     sa.add_at(i1, v1);
498     /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
499     /*: ghost specvar sa__csize :: "int" = "sa..csize" */
500     /*: assume "0 <= i2 & i2 < sa..csize" */
501     sa.remove_at(i2);
502     /*: assume "~((i1 < i2 & i2 < sa..csize & (ALL v. ((i2, v) : sa__contents) =
503     ((i2, v) : sa..contents)) & 0 <= i2 & i2 < sa__csize & 0 <= i2 & i2 <
504     sa..csize) | (sa..csize - 1 >= i1 & i1 = i2 & (i1, v1) : sa..contents & 0 <=
505     i1 & i1 < sa..csize) | (sa..csize - 1 >= i1 & i1 > i2 & (i1, v1) :
506     sa..contents & 0 <= i1 & i1 < sa..csize))" */
507
508     /*: assume "0 <= i2 & i2 < sb..csize" */
509     sb.remove_at(i2);
510     /*: assume "0 <= i1 & i1 <= sb..csize" */
511     sb.add_at(i1, v1);
512
513     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
514 }
515
516 static void add_at_set_pre_s_18(ArrayList sa, ArrayList sb, int i1, Object v1, int
517     i2, Object v2)
518 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
519     sa..contents = sb..contents & sa..csize = sb..csize"
520     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
521     "sb..msize"
522     ensures "True" */
523 {
524     /*: assume "(i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents) =
525     ((i2, v) : sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) :
526     sa..contents & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize)
527     | (i1 = i2 & i2 < sa..csize & (i2, v1) : sa..contents & (i2, v2) :
528     sa..contents & v1 = v2 & 0 <= i2 & i2 < sa..csize) | i1 > i2" */
529     /*: assume "0 <= i1 & i1 <= sa..csize" */
530     sa.add_at(i1, v1);
531     /*: assume "0 <= i2 & i2 < sa..csize" */
532     Object r2a = sa.set(i2, v2);
533
534     Object r2b = sb.set(i2, v2);
535     sb.add_at(i1, v1);
536
537     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
538 }
539
540 static void add_at_set_pre_c_18(ArrayList sa, ArrayList sb, int i1, Object v1, int
541     i2, Object v2)
542 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
543     sa..contents = sb..contents & sa..csize = sb..csize"
544     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
545     "sb..msize"
546     ensures "True" */
547 {
548     /*: assume "~((i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents) =
549     ((i2, v) : sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) :
550     sa..contents & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize)
551     | (i1 = i2 & i2 < sa..csize & (i2, v1) : sa..contents & (i2, v2) :
552     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,
549     int i2, Object v2)
550 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
551     sa..contents = sb..contents & sa..csize = sb..csize"
552     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
553     "sb..msize"
554     ensures "True" */
555 {
556     /*: assume "0 <= i1 & i1 <= sa..csize" */
557     sa.add_at(i1, v1);
558     /*: assume "(i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa..contents) =
559     ((i2 + 1, v) : sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) :
560     sa..contents & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize)
561     | (i1 = i2 & i2 < sa..csize - 1 & (i2 + 1, v1) : sa..contents & (i2 + 1, v2)
562     : sa..contents & v1 = v2 & 0 <= i2 + 1 & i2 + 1 < sa..csize) | i1 > i2" */
563     /*: assume "0 <= i2 & i2 < sa..csize" */
564     Object r2a = sa.set(i2, v2);
565
566     Object r2b = sb.set(i2, v2);
567     sb.add_at(i1, v1);
568
569     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
570 }
571
572 static void add_at_set_between_c_19(ArrayList sa, ArrayList sb, int i1, Object v1,
573     int i2, Object v2)
574 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
575     sa..contents = sb..contents & sa..csize = sb..csize"
576     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
577     "sb..msize"
578     ensures "True" */
579 {
580     /*: assume "0 <= i1 & i1 <= sa..csize" */
581     sa.add_at(i1, v1);
582     /*: assume "~((i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa..contents)
583     = ((i2 + 1, v) : sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) :
584     sa..contents & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize)
585     | (i1 = i2 & i2 < sa..csize - 1 & (i2 + 1, v1) : sa..contents & (i2 + 1, v2)
586     : sa..contents & v1 = v2 & 0 <= i2 + 1 & i2 + 1 < sa..csize) | i1 > i2)" */
587     /*: assume "0 <= i2 & i2 < sa..csize" */
588     Object r2a = sa.set(i2, v2);
589
590     /*: assume "0 <= i2 & i2 < sb..csize" */
591     Object r2b = sb.set(i2, v2);
592     /*: assume "0 <= i1 & i1 <= sb..csize" */
593     sb.add_at(i1, v1);
594
595     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
596 }

```

```

586 static void add_at_set_post_s_20(ArrayList sa, ArrayList sb, int i1, Object v1, int
    i2, Object v2)
587 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
588     sa..contents = sb..contents & sa..csize = sb..csize"
589     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
        "sb..msize"
590     ensures "True" */
591 {
592     /*: assume "0 <= i1 & i1 <= sa..csize" */
593     sa.add_at(i1, v1);
594     /*: assume "0 <= i2 & i2 < sa..csize" */
595     Object r2a = sa.set(i2, v2);
596     /*: 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" */
597
598     Object r2b = sb.set(i2, v2);
599     sb.add_at(i1, v1);
600
601     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
602 }
603
604 static void add_at_set_post_c_20(ArrayList sa, ArrayList sb, int i1, Object v1, int
    i2, Object v2)
605 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
606     sa..contents = sb..contents & sa..csize = sb..csize"
607     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
        "sb..msize"
608     ensures "True" */
609 {
610     /*: assume "0 <= i1 & i1 <= sa..csize" */
611     sa.add_at(i1, v1);
612     /*: assume "0 <= i2 & i2 < sa..csize" */
613     Object r2a = sa.set(i2, v2);
614     /*: 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)" */
615
616     /*: assume "0 <= i2 & i2 < sb..csize" */
617     Object r2b = sb.set(i2, v2);
618     /*: assume "0 <= i1 & i1 <= sb..csize" */
619     sb.add_at(i1, v1);
620
621     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
622 }
623
624 static void add_at_set_pre_s_21(ArrayList sa, ArrayList sb, int i1, Object v1, int
    i2, Object v2)
625 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
626     sa..contents = sb..contents & sa..csize = sb..csize"
627     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
        "sb..msize"
628     ensures "True" */
629 {
630     /*: 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" */
631     /*: assume "0 <= i1 & i1 <= sa..csize" */
632     sa.add_at(i1, v1);
633     /*: 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,
643     int i2, Object v2)
644 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
645     sa..contents = sb..contents & sa..csize = sb..csize"
646     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
647     "sb..msize"
648     ensures "True" */
649 {
650     /*: assume "0 <= i1 & i1 <= sa..csize" */
651     sa.add_at(i1, v1);
652     /*: assume "(i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa..contents) =
653     ((i2 + 1, v) : sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) :
654     sa..contents & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize)
655     | (i1 = i2 & i2 < sa..csize - 1 & (i2 + 1, v1) : sa..contents & (i2 + 1, v2)
656     : sa..contents & v1 = v2 & 0 <= i2 + 1 & i2 + 1 < sa..csize) | i1 > i2" */
657     /*: assume "0 <= i2 & i2 < sa..csize" */
658     sa.set(i2, v2);
659
660     sb.set(i2, v2);
661     sb.add_at(i1, v1);
662
663     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
664 }
665
666 static void add_at_set_between_c_22(ArrayList sa, ArrayList sb, int i1, Object v1,
667     int i2, Object v2)
668 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
669     sa..contents = sb..contents & sa..csize = sb..csize"
670     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
671     "sb..msize"
672     ensures "True" */
673 {
674     /*: assume "0 <= i1 & i1 <= sa..csize" */
675     sa.add_at(i1, v1);
676     /*: assume "~((i1 < i2 & i2 < sa..csize - 1 & (ALL v. ((i2, v) : sa..contents)
677     = ((i2 + 1, v) : sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) :
678     sa..contents & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize)
679     | (i1 = i2 & i2 < sa..csize - 1 & (i2 + 1, v1) : sa..contents & (i2 + 1, v2)
680     : sa..contents & v1 = v2 & 0 <= i2 + 1 & i2 + 1 < sa..csize) | i1 > i2)" */
681     /*: assume "0 <= i2 & i2 < sa..csize" */
682     sa.set(i2, v2);
683
684     /*: assume "0 <= i2 & i2 < sb..csize" */
685     sb.set(i2, v2);
686     /*: assume "0 <= i1 & i1 <= sb..csize" */
687     sb.add_at(i1, v1);
688
689     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
690 }
691
692 static void add_at_set_post_s_23(ArrayList sa, ArrayList sb, int i1, Object v1, int
693     i2, Object v2)
694 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
695     sa..contents = sb..contents & sa..csize = sb..csize"
696     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
697     "sb..msize"
698     ensures "True" */
699

```



```

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",
704     "sb..msize"
705     ensures "True" */
706 {
707     /*: assume "0 <= i1 & i1 <= sa..csize" */
708     sa.add_at(i1, v1);
709     /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
710     /*: ghost specvar sa__csize :: "int" = "sa..csize" */
711     /*: assume "0 <= i2 & i2 < sa..csize" */
712     sa.set(i2, v2);
713     /*: 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)" */
714
715     /*: assume "0 <= i2 & i2 < sb..csize" */
716     sb.set(i2, v2);
717     /*: assume "0 <= i1 & i1 <= sb..csize" */
718     sb.add_at(i1, v1);
719
720     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
721 }
722
723 static void add_at_size_pre_s_24(ArrayList sa, ArrayList sb, int i1, Object v1)
724 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
725     sa..contents = sb..contents & sa..csize = sb..csize"
726     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
727     "sb..msize"
728     ensures "True" */
729 {
730     /*: assume "False" */
731     /*: assume "0 <= i1 & i1 <= sa..csize" */
732     sa.add_at(i1, v1);
733     int r2a = sa.size();
734
735     int r2b = sb.size();
736     sb.add_at(i1, v1);
737
738     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
739 }

```

```

739 static void add_at_size_pre_c_24(ArrayList sa, ArrayList sb, int i1, Object v1)
740 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
741      sa..contents = sb..contents & sa..csize = sb..csize"
742      modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
743      "sb..msize"
744      ensures "True" */
745 {
746     /*: assume "~(False)" */
747     /*: assume "0 <= i1 & i1 <= sa..csize" */
748     sa.add_at(i1, v1);
749     int r2a = sa.size();
750
751     int r2b = sb.size();
752     /*: assume "0 <= i1 & i1 <= sb..csize" */
753     sb.add_at(i1, v1);
754
755     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
756     */
757 }
758
759 static void add_at_size_between_s_25(ArrayList sa, ArrayList sb, int i1, Object v1)
760 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
761      sa..contents = sb..contents & sa..csize = sb..csize"
762      modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
763      "sb..msize"
764      ensures "True" */
765 {
766     /*: assume "0 <= i1 & i1 <= sa..csize" */
767     sa.add_at(i1, v1);
768     /*: assume "False" */
769     int r2a = sa.size();
770
771     int r2b = sb.size();
772     sb.add_at(i1, v1);
773
774     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
775 }
776
777 static void add_at_size_between_c_25(ArrayList sa, ArrayList sb, int i1, Object v1)
778 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
779      sa..contents = sb..contents & sa..csize = sb..csize"
780      modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
781      "sb..msize"
782      ensures "True" */
783 {
784     /*: assume "0 <= i1 & i1 <= sa..csize" */
785     sa.add_at(i1, v1);
786     /*: assume "~(False)" */
787     int r2a = sa.size();
788
789     int r2b = sb.size();
790     /*: assume "0 <= i1 & i1 <= sb..csize" */
791     sb.add_at(i1, v1);
792
793     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
794     */
795 }
796
797 static void add_at_size_post_s_26(ArrayList sa, ArrayList sb, int i1, Object v1)
798 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
799      sa..contents = sb..contents & sa..csize = sb..csize"
800      modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
801      "sb..msize"
802      ensures "True" */
803 {

```

```

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",
813     "sb..msize"
814     ensures "True" */
815 {
816     /*: assume "0 <= i1 & i1 <= sa..csize" */
817     sa.add_at(i1, v1);
818     int r2a = sa.size();
819     /*: assume "~(False)" */
820
821     int r2b = sb.size();
822     /*: assume "0 <= i1 & i1 <= sb..csize" */
823     sb.add_at(i1, v1);
824
825     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
826     */
827 }
828
829 static void get_add_at_pre_s_27(ArrayList sa, ArrayList sb, int i1, int i2, Object
830     v2)
831 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
832     sa..contents = sb..contents & sa..csize = sb..csize"
833     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
834     "sb..msize"
835     ensures "True" */
836 {
837     /*: assume "i1 < i2 | (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 <
838     sa..csize) | (i1 > i2 & (ALL v. ((i1 - 1, v) : sa..contents) = ((i1, v) :
839     sa..contents)) & 0 <= i1 - 1 & i1 - 1 < sa..csize & 0 <= i1 & i1 <
840     sa..csize)" */
841     /*: assume "0 <= i1 & i1 < sa..csize" */
842     Object r1a = sa.get(i1);
843     /*: assume "0 <= i2 & i2 <= sa..csize" */
844     sa.add_at(i2, v2);
845
846     sb.add_at(i2, v2);
847     Object r1b = sb.get(i1);
848
849     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
850 }
851
852 static void get_add_at_pre_c_27(ArrayList sa, ArrayList sb, int i1, int i2, Object
853     v2)
854 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
855     sa..contents = sb..contents & sa..csize = sb..csize"
856     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
857     "sb..msize"
858     ensures "True" */
859 {
860     /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 <
861     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))" */
852 /*: assume "0 <= i1 & i1 < sa..csize" */
853 Object r1a = sa.get(i1);
854 /*: assume "0 <= i2 & i2 <= sa..csize" */
855 sa.add_at(i2, v2);
856
857 /*: assume "0 <= i2 & i2 <= sb..csize" */
858 sb.add_at(i2, v2);
859 /*: assume "0 <= i1 & i1 < sb..csize" */
860 Object r1b = sb.get(i1);
861
862 /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
    */
863 }
864
865 static void get_add_at_between_s_28(ArrayList sa, ArrayList sb, int i1, int i2,
    Object v2)
866 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
867     sa..contents = sb..contents & sa..csize = sb..csize"
868     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
869     "sb..msize"
870     ensures "True" */
871 {
872     /*: assume "0 <= i1 & i1 < sa..csize" */
873     Object r1a = sa.get(i1);
874     /*: assume "i1 < i2 | (i1 = i2 & r1a = v2) | (i1 > i2 & (i1 - 1, r1a) :
875     sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize)" */
876     /*: assume "0 <= i2 & i2 <= sa..csize" */
877     sa.add_at(i2, v2);
878
879     sb.add_at(i2, v2);
880     Object r1b = sb.get(i1);
881
882     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
883 }
884
885 static void get_add_at_between_c_28(ArrayList sa, ArrayList sb, int i1, int i2,
    Object v2)
886 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
887     sa..contents = sb..contents & sa..csize = sb..csize"
888     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
889     "sb..msize"
890     ensures "True" */
891 {
892     /*: assume "0 <= i1 & i1 < sa..csize" */
893     Object r1a = sa.get(i1);
894     /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v2) | (i1 > i2 & (i1 - 1, r1a) :
895     sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize))" */
896     /*: assume "0 <= i2 & i2 <= sa..csize" */
897     sa.add_at(i2, v2);
898
899     /*: assume "0 <= i2 & i2 <= sb..csize" */
900     sb.add_at(i2, v2);
901     /*: assume "0 <= i1 & i1 < sb..csize" */
902     Object r1b = sb.get(i1);
903
904     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
905     */
906 }
907
908 static void get_add_at_post_s_29(ArrayList sa, ArrayList sb, int i1, int i2, Object
    v2)
909 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
910     sa..contents = sb..contents & sa..csize = sb..csize"

```

```

906     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
907           "sb..msize"
908     ensures "True" */
909 {
910     /*: assume "0 <= i1 & i1 < sa..csize" */
911     Object r1a = sa.get(i1);
912     /*: assume "0 <= i2 & i2 <= sa..csize" */
913     sa.add_at(i2, v2);
914     /*: assume "i1 < i2 | (i1 = i2 & r1a = v2) | (i1 > i2 & (i1, r1a) :
915           sa..contents & 0 <= i1 & i1 < sa..csize)" */
916     sb.add_at(i2, v2);
917     Object r1b = sb.get(i1);
918     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
919 }
920
921 static void get_add_at_post_c_29(ArrayList sa, ArrayList sb, int i1, int i2, Object
922     v2)
923 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
924     sa..contents = sb..contents & sa..csize = sb..csize"
925     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
926     "sb..msize"
927     ensures "True" */
928 {
929     /*: assume "0 <= i1 & i1 < sa..csize" */
930     Object r1a = sa.get(i1);
931     /*: assume "0 <= i2 & i2 <= sa..csize" */
932     sa.add_at(i2, v2);
933     /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v2) | (i1 > i2 & (i1, r1a) :
934           sa..contents & 0 <= i1 & i1 < sa..csize))" */
935     /*: assume "0 <= i2 & i2 <= sb..csize" */
936     sb.add_at(i2, v2);
937     /*: assume "0 <= i1 & i1 < sb..csize" */
938     Object r1b = sb.get(i1);
939     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
940           */
941 }
942
943 static void get_get_pre_s_30(ArrayList sa, ArrayList sb, int i1, int i2)
944 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
945     sa..contents = sb..contents & sa..csize = sb..csize"
946     ensures "True" */
947 {
948     /*: assume "True" */
949     /*: assume "0 <= i1 & i1 < sa..csize" */
950     Object r1a = sa.get(i1);
951     /*: assume "0 <= i2 & i2 < sa..csize" */
952     Object r2a = sa.get(i2);
953     Object r2b = sb.get(i2);
954     Object r1b = sb.get(i1);
955     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
956           sb..csize" */
957 }
958
959 static void get_get_pre_c_30(ArrayList sa, ArrayList sb, int i1, int i2)
960 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
961     sa..contents = sb..contents & sa..csize = sb..csize"
962     ensures "True" */
963 {
964     /*: 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     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1027         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 =
1047         sb..csize)" */
1048 }
1049
1050 static void get_indexOf_pre_s_33(ArrayList sa, ArrayList sb, int i1, Object v2)
1051 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1052     sa..contents = sb..contents & sa..csize = sb..csize"
1053     ensures "True" */
1054 {
1055     /*: assume "True" */
1056     /*: assume "0 <= i1 & i1 < sa..csize" */
1057     Object r1a = sa.get(i1);
1058     int r2a = sa.indexOf(v2);
1059
1060     int r2b = sb.indexOf(v2);
1061     Object r1b = sb.get(i1);
1062
1063     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1064         sb..csize" */
1065 }
1066
1067 static void get_indexOf_pre_c_33(ArrayList sa, ArrayList sb, int i1, Object v2)
1068 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1069     sa..contents = sb..contents & sa..csize = sb..csize"
1070     ensures "True" */
1071 {
1072     /*: assume "~(True)" */
1073     /*: assume "0 <= i1 & i1 < sa..csize" */
1074     Object r1a = sa.get(i1);
1075     int r2a = sa.indexOf(v2);
1076
1077     int r2b = sb.indexOf(v2);
1078     /*: assume "0 <= i1 & i1 < sb..csize" */
1079     Object r1b = sb.get(i1);
1080
1081     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1082         sb..csize)" */
1083 }
1084
1085 static void get_indexOf_between_s_34(ArrayList sa, ArrayList sb, int i1, Object v2)
1086 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1087     sa..contents = sb..contents & sa..csize = sb..csize"
1088     ensures "True" */
1089 {

```

```

1087     /*: assume "0 <= i1 & i1 < sa..csize" */
1088     Object r1a = sa.get(i1);
1089     /*: assume "True" */
1090     int r2a = sa.indexOf(v2);
1091
1092     int r2b = sb.indexOf(v2);
1093     Object r1b = sb.get(i1);
1094
1095     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1096         sb..csize" */
1097 }
1098
1099 static void get_indexOf_between_c_34(ArrayList sa, ArrayList sb, int i1, Object v2)
1100 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1101     sa..contents = sb..contents & sa..csize = sb..csize"
1102     ensures "True" */
1103 {
1104     /*: assume "0 <= i1 & i1 < sa..csize" */
1105     Object r1a = sa.get(i1);
1106     /*: assume "~(True)" */
1107     int r2a = sa.indexOf(v2);
1108
1109     int r2b = sb.indexOf(v2);
1110     /*: assume "0 <= i1 & i1 < sb..csize" */
1111     Object r1b = sb.get(i1);
1112
1113     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1114         sb..csize)" */
1115 }
1116
1117 static void get_indexOf_post_s_35(ArrayList sa, ArrayList sb, int i1, Object v2)
1118 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1119     sa..contents = sb..contents & sa..csize = sb..csize"
1120     ensures "True" */
1121 {
1122     /*: assume "0 <= i1 & i1 < sa..csize" */
1123     Object r1a = sa.get(i1);
1124     int r2a = sa.indexOf(v2);
1125     /*: assume "True" */
1126
1127     int r2b = sb.indexOf(v2);
1128     Object r1b = sb.get(i1);
1129
1130     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1131         sb..csize" */
1132 }
1133
1134 static void get_indexOf_post_c_35(ArrayList sa, ArrayList sb, int i1, Object v2)
1135 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1136     sa..contents = sb..contents & sa..csize = sb..csize"
1137     ensures "True" */
1138 {
1139     /*: assume "0 <= i1 & i1 < sa..csize" */
1140     Object r1a = sa.get(i1);
1141     int r2a = sa.indexOf(v2);
1142     /*: assume "~(True)" */
1143
1144     int r2b = sb.indexOf(v2);
1145     /*: assume "0 <= i1 & i1 < sb..csize" */
1146     Object r1b = sb.get(i1);
1147
1148     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1149         sb..csize)" */
1150 }

```



```

1148 static void get_lastIndex0f_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.lastIndex0f(v2);
1157
1158     int r2b = sb.lastIndex0f(v2);
1159     Object r1b = sb.get(i1);
1160
1161     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1162             sb..csize" */
1163 }
1164
1164 static void get_lastIndex0f_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.lastIndex0f(v2);
1173
1174     int r2b = sb.lastIndex0f(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 =
1179             sb..csize)" */
1180 }
1181
1181 static void get_lastIndex0f_between_s_37(ArrayList sa, ArrayList sb, int i1, Object
1182 v2)
1183 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1184          sa..contents = sb..contents & sa..csize = sb..csize"
1185          ensures "True" */
1186 {
1187     /*: assume "0 <= i1 & i1 < sa..csize" */
1188     Object r1a = sa.get(i1);
1189     /*: assume "True" */
1190     int r2a = sa.lastIndex0f(v2);
1191
1192     int r2b = sb.lastIndex0f(v2);
1193     Object r1b = sb.get(i1);
1194
1195     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1196             sb..csize" */
1197 }
1198
1198 static void get_lastIndex0f_between_c_37(ArrayList sa, ArrayList sb, int i1, Object
1199 v2)
1200 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1201          sa..contents = sb..contents & sa..csize = sb..csize"
1202          ensures "True" */
1203 {
1204     /*: assume "0 <= i1 & i1 < sa..csize" */
1205     Object r1a = sa.get(i1);
1206     /*: assume "~(True)" */
1207     int r2a = sa.lastIndex0f(v2);
1208
1209     int r2b = sb.lastIndex0f(v2);

```

```

1208     /*: assume "0 <= i1 & i1 < sb..csize" */
1209     Object r1b = sb.get(i1);
1210
1211     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1212         sb..csize)" */
1213 }
1214
1215 static void get_lastIndexof_post_s_38(ArrayList sa, ArrayList sb, int i1, Object v2)
1216 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1217     sa..contents = sb..contents & sa..csize = sb..csize"
1218     ensures "True" */
1219 {
1220     /*: assume "0 <= i1 & i1 < sa..csize" */
1221     Object r1a = sa.get(i1);
1222     int r2a = sa.lastIndexof(v2);
1223     /*: assume "True" */
1224
1225     int r2b = sb.lastIndexof(v2);
1226     Object r1b = sb.get(i1);
1227
1228     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1229         sb..csize" */
1230 }
1231
1232 static void get_lastIndexof_post_c_38(ArrayList sa, ArrayList sb, int i1, Object v2)
1233 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1234     sa..contents = sb..contents & sa..csize = sb..csize"
1235     ensures "True" */
1236 {
1237     /*: assume "0 <= i1 & i1 < sa..csize" */
1238     Object r1a = sa.get(i1);
1239     int r2a = sa.lastIndexof(v2);
1240     /*: assume "~(True)" */
1241
1242     int r2b = sb.lastIndexof(v2);
1243     /*: assume "0 <= i1 & i1 < sb..csize" */
1244     Object r1b = sb.get(i1);
1245
1246     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1247         sb..csize)" */
1248 }
1249
1250 static void get_remove_at_pre_s_39(ArrayList sa, ArrayList sb, int i1, int i2)
1251 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1252     sa..contents = sb..contents & sa..csize = sb..csize"
1253     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1254     ensures "True" */
1255 {
1256     /*: assume "i1 < i2 | (sa..csize - 1 > i1 & i1 = i2 & (ALL v. ((i1, v) :
1257         sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
1258         <= i1 + 1 & i1 + 1 < sa..csize | (sa..csize - 1 > i1 & i1 > i2 & (ALL v.
1259         ((i1, v) : sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 <
1260         sa..csize & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
1261     /*: assume "0 <= i1 & i1 < sa..csize" */
1262     Object r1a = sa.get(i1);
1263     /*: assume "0 <= i2 & i2 < sa..csize" */
1264     Object r2a = sa.remove_at(i2);
1265
1266     Object r2b = sb.remove_at(i2);
1267     Object r1b = sb.get(i1);
1268
1269     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1270         sb..csize" */
1271 }

```

```

1265 static void get_remove_at_pre_c_39(ArrayList sa, ArrayList sb, int i1, int i2)
1266 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1267      sa..contents = sb..contents & sa..csize = sb..csize"
1268 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1269 ensures "True" */
1270 {
1271     /*: assume "~(i1 < i2 | (sa..csize - 1 > i1 & i1 = i2 & (ALL v. ((i1, v) :
1272      sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
1273      <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v.
1274      ((i1, v) : sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 <
1275      sa..csize & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
1276     /*: assume "0 <= i1 & i1 < sa..csize" */
1277     Object r1a = sa.get(i1);
1278     /*: assume "0 <= i2 & i2 < sa..csize" */
1279     Object r2a = sa.remove_at(i2);
1280
1281     /*: assume "0 <= i2 & i2 < sb..csize" */
1282     Object r2b = sb.remove_at(i2);
1283     /*: assume "0 <= i1 & i1 < sb..csize" */
1284     Object r1b = sb.get(i1);
1285
1286     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1287      sb..csize)" */
1288 }
1289
1290 static void get_remove_at_between_s_40(ArrayList sa, ArrayList sb, int i1, int i2)
1291 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1292      sa..contents = sb..contents & sa..csize = sb..csize"
1293 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1294 ensures "True" */
1295 {
1296     /*: assume "0 <= i1 & i1 < sa..csize" */
1297     Object r1a = sa.get(i1);
1298     /*: assume "i1 < i2 | (sa..csize - 1 > i1 & i1 = i2 & (i1 + 1, r1a) :
1299      sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
1300      > i2 & (i1 + 1, r1a) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
1301     /*: assume "0 <= i2 & i2 < sa..csize" */
1302     Object r2a = sa.remove_at(i2);
1303
1304     Object r2b = sb.remove_at(i2);
1305     Object r1b = sb.get(i1);
1306
1307     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1308      sb..csize" */
1309 }
1310
1311 static void get_remove_at_between_c_40(ArrayList sa, ArrayList sb, int i1, int i2)
1312 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1313      sa..contents = sb..contents & sa..csize = sb..csize"
1314 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1315 ensures "True" */
1316 {
1317     /*: assume "0 <= i1 & i1 < sa..csize" */
1318     Object r1a = sa.get(i1);
1319     /*: assume "~(i1 < i2 | (sa..csize - 1 > i1 & i1 = i2 & (i1 + 1, r1a) :
1320      sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
1321      > i2 & (i1 + 1, r1a) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
1322     /*: assume "0 <= i2 & i2 < sa..csize" */
1323     Object r2a = sa.remove_at(i2);
1324
1325     /*: assume "0 <= i2 & i2 < sb..csize" */
1326     Object r2b = sb.remove_at(i2);
1327     /*: assume "0 <= i1 & i1 < sb..csize" */
1328     Object r1b = sb.get(i1);
1329 }

```

```

1320     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1321         sb..csize)" */
1322 }
1323
1324 static void get_remove_at_post_s_41(ArrayList sa, ArrayList sb, int i1, int i2)
1325 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1326     sa..contents = sb..contents & sa..csize = sb..csize"
1327     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1328     ensures "True" */
1329 {
1330     /*: assume "0 <= i1 & i1 < sa..csize" */
1331     Object r1a = sa.get(i1);
1332     /*: assume "0 <= i2 & i2 < sa..csize" */
1333     Object r2a = sa.remove_at(i2);
1334     /*: assume "i1 < i2 | (sa..csize > i1 & i1 = i2 & (i1, r1a) : sa..contents & 0
1335         <= i1 & i1 < sa..csize) | (sa..csize > i1 & i1 > i2 & (i1, r1a) :
1336         sa..contents & 0 <= i1 & i1 < sa..csize)" */
1337
1338     Object r2b = sb.remove_at(i2);
1339     Object r1b = sb.get(i1);
1340
1341     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1342         sb..csize" */
1343 }
1344
1345 static void get_remove_at_post_c_41(ArrayList sa, ArrayList sb, int i1, int i2)
1346 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1347     sa..contents = sb..contents & sa..csize = sb..csize"
1348     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1349     ensures "True" */
1350 {
1351     /*: assume "0 <= i1 & i1 < sa..csize" */
1352     Object r1a = sa.get(i1);
1353     /*: assume "0 <= i2 & i2 < sa..csize" */
1354     Object r2a = sa.remove_at(i2);
1355     /*: assume "~(i1 < i2 | (sa..csize > i1 & i1 = i2 & (i1, r1a) : sa..contents &
1356         0 <= i1 & i1 < sa..csize) | (sa..csize > i1 & i1 > i2 & (i1, r1a) :
1357         sa..contents & 0 <= i1 & i1 < sa..csize))" */
1358
1359     /*: assume "0 <= i2 & i2 < sb..csize" */
1360     Object r2b = sb.remove_at(i2);
1361     /*: assume "0 <= i1 & i1 < sb..csize" */
1362     Object r1b = sb.get(i1);
1363
1364     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1365         sb..csize)" */
1366 }
1367
1368 static void get_remove_at_pre_s_42(ArrayList sa, ArrayList sb, int i1, int i2)
1369 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1370     sa..contents = sb..contents & sa..csize = sb..csize"
1371     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1372     ensures "True" */
1373 {
1374     /*: assume "i1 < i2 | (sa..csize - 1 > i1 & i1 = i2 & (ALL v. ((i1, v) :
1375         sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
1376         <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v.
1377         ((i1, v) : sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 <
1378         sa..csize & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
1379     /*: assume "0 <= i1 & i1 < sa..csize" */
1380     Object r1a = sa.get(i1);
1381     /*: assume "0 <= i2 & i2 < sa..csize" */
1382     sa.remove_at(i2);
1383
1384     sb.remove_at(i2);

```

```

1374     Object r1b = sb.get(i1);
1375
1376     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
1377 }
1378
1379 static void get_remove_at_pre_c_42(ArrayList sa, ArrayList sb, int i1, int i2)
1380 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1381     sa..contents = sb..contents & sa..csize = sb..csize"
1382     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1383     ensures "True" */
1384 {
1385     /*: assume "~(i1 < i2 | (sa..csize - 1 > i1 & i1 = i2 & (ALL v. ((i1, v) :
1386     sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
1387     <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v.
1388     ((i1, v) : sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 <
1389     sa..csize & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
1390     /*: assume "0 <= i1 & i1 < sa..csize" */
1391     Object r1a = sa.get(i1);
1392     /*: assume "0 <= i2 & i2 < sa..csize" */
1393     sa.remove_at(i2);
1394
1395     /*: assume "0 <= i2 & i2 < sb..csize" */
1396     sb.remove_at(i2);
1397     /*: assume "0 <= i1 & i1 < sb..csize" */
1398     Object r1b = sb.get(i1);
1399
1400     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
1401     */
1402 }
1403
1404 static void get_remove_at_between_s_43(ArrayList sa, ArrayList sb, int i1, int i2)
1405 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1406     sa..contents = sb..contents & sa..csize = sb..csize"
1407     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1408     ensures "True" */
1409 {
1410     /*: assume "0 <= i1 & i1 < sa..csize" */
1411     Object r1a = sa.get(i1);
1412     /*: assume "i1 < i2 | (sa..csize - 1 > i1 & i1 = i2 & (i1 + 1, r1a) :
1413     sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
1414     > i2 & (i1 + 1, r1a) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
1415     /*: assume "0 <= i2 & i2 < sa..csize" */
1416     sa.remove_at(i2);
1417
1418     sb.remove_at(i2);
1419     Object r1b = sb.get(i1);
1420
1421     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
1422 }
1423
1424 static void get_remove_at_between_c_43(ArrayList sa, ArrayList sb, int i1, int i2)
1425 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1426     sa..contents = sb..contents & sa..csize = sb..csize"
1427     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1428     ensures "True" */
1429 {
1430     /*: assume "0 <= i1 & i1 < sa..csize" */
1431     Object r1a = sa.get(i1);
1432     /*: assume "~(i1 < i2 | (sa..csize - 1 > i1 & i1 = i2 & (i1 + 1, r1a) :
1433     sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
1434     > i2 & (i1 + 1, r1a) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
1435     /*: assume "0 <= i2 & i2 < sa..csize" */
1436     sa.remove_at(i2);
1437
1438     /*: assume "0 <= i2 & i2 < sb..csize" */

```

```

1430     sb.remove_at(i2);
1431     /*: assume "0 <= i1 & i1 < sb..csize" */
1432     Object r1b = sb.get(i1);
1433
1434     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
1435 }
1436
1437 static void get_remove_at_post_s_44(ArrayList sa, ArrayList sb, int i1, int i2)
1438 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1439     sa..contents = sb..contents & sa..csize = sb..csize"
1440 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1441 ensures "True" */
1442 {
1443     /*: assume "0 <= i1 & i1 < sa..csize" */
1444     Object r1a = sa.get(i1);
1445     /*: assume "0 <= i2 & i2 < sa..csize" */
1446     sa.remove_at(i2);
1447     /*: 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)" */
1448
1449     sb.remove_at(i2);
1450     Object r1b = sb.get(i1);
1451
1452     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
1453 }
1454
1455 static void get_remove_at_post_c_44(ArrayList sa, ArrayList sb, int i1, int i2)
1456 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1457     sa..contents = sb..contents & sa..csize = sb..csize"
1458 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1459 ensures "True" */
1460 {
1461     /*: assume "0 <= i1 & i1 < sa..csize" */
1462     Object r1a = sa.get(i1);
1463     /*: assume "0 <= i2 & i2 < sa..csize" */
1464     sa.remove_at(i2);
1465     /*: 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))" */
1466
1467     /*: assume "0 <= i2 & i2 < sb..csize" */
1468     sb.remove_at(i2);
1469     /*: assume "0 <= i1 & i1 < sb..csize" */
1470     Object r1b = sb.get(i1);
1471
1472     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
1473 }
1474
1475 static void get_set_pre_s_45(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
1476 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1477     sa..contents = sb..contents & sa..csize = sb..csize"
1478 modifies "sa..contents", "sb..contents"
1479 ensures "True" */
1480 {
1481     /*: assume "i1 < i2 | (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 <
        sa..csize) | i1 > i2" */
1482     /*: assume "0 <= i1 & i1 < sa..csize" */
1483     Object r1a = sa.get(i1);
1484     /*: assume "0 <= i2 & i2 < sa..csize" */
1485     Object r2a = sa.set(i2, v2);
1486
1487     Object r2b = sb.set(i2, v2);

```

```

1488     Object r1b = sb.get(i1);
1489
1490     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1491         sb..csize" */
1492 }
1493
1494 static void get_set_pre_c_45(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
1495 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1496     sa..contents = sb..contents & sa..csize = sb..csize"
1497     modifies "sa..contents", "sb..contents"
1498     ensures "True" */
1499 {
1500     /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 <
1501         sa..csize) | i1 > i2)" */
1502     /*: assume "0 <= i1 & i1 < sa..csize" */
1503     Object r1a = sa.get(i1);
1504     /*: assume "0 <= i2 & i2 < sa..csize" */
1505     Object r2a = sa.set(i2, v2);
1506
1507     /*: assume "0 <= i2 & i2 < sb..csize" */
1508     Object r2b = sb.set(i2, v2);
1509     /*: assume "0 <= i1 & i1 < sb..csize" */
1510     Object r1b = sb.get(i1);
1511
1512     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1513         sb..csize)" */
1514 }
1515
1516 static void get_set_between_s_46(ArrayList sa, ArrayList sb, int i1, int i2, Object
1517     v2)
1518 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1519     sa..contents = sb..contents & sa..csize = sb..csize"
1520     modifies "sa..contents", "sb..contents"
1521     ensures "True" */
1522 {
1523     /*: assume "0 <= i1 & i1 < sa..csize" */
1524     Object r1a = sa.get(i1);
1525     /*: assume "i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2" */
1526     /*: assume "0 <= i2 & i2 < sa..csize" */
1527     Object r2a = sa.set(i2, v2);
1528
1529     Object r2b = sb.set(i2, v2);
1530     Object r1b = sb.get(i1);
1531
1532     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1533         sb..csize" */
1534 }
1535
1536 static void get_set_between_c_46(ArrayList sa, ArrayList sb, int i1, int i2, Object
1537     v2)
1538 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1539     sa..contents = sb..contents & sa..csize = sb..csize"
1540     modifies "sa..contents", "sb..contents"
1541     ensures "True" */
1542 {
1543     /*: assume "0 <= i1 & i1 < sa..csize" */
1544     Object r1a = sa.get(i1);
1545     /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2)" */
1546     /*: assume "0 <= i2 & i2 < sa..csize" */
1547     Object r2a = sa.set(i2, v2);
1548
1549     /*: assume "0 <= i2 & i2 < sb..csize" */
1550     Object r2b = sb.set(i2, v2);
1551     /*: assume "0 <= i1 & i1 < sb..csize" */
1552     Object r1b = sb.get(i1);

```

```

1547     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1548         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 =
1567         sb..csize" */
1568 }
1569
1570 static void get_set_post_c_47(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
1571 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1572     sa..contents = sb..contents & sa..csize = sb..csize"
1573     modifies "sa..contents", "sb..contents"
1574     ensures "True" */
1575 {
1576     /*: assume "0 <= i1 & i1 < sa..csize" */
1577     Object r1a = sa.get(i1);
1578     /*: assume "0 <= i2 & i2 < sa..csize" */
1579     Object r2a = sa.set(i2, v2);
1580     /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2)" */
1581
1582     /*: assume "0 <= i2 & i2 < sb..csize" */
1583     Object r2b = sb.set(i2, v2);
1584     /*: assume "0 <= i1 & i1 < sb..csize" */
1585     Object r1b = sb.get(i1);
1586
1587     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1588         sb..csize)" */
1589 }
1590
1591 static void get_set_pre_s_48(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
1592 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1593     sa..contents = sb..contents & sa..csize = sb..csize"
1594     modifies "sa..contents", "sb..contents"
1595     ensures "True" */
1596 {
1597     /*: assume "i1 < i2 | (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 <
1598         sa..csize) | i1 > i2" */
1599     /*: assume "0 <= i1 & i1 < sa..csize" */
1600     Object r1a = sa.get(i1);
1601     /*: assume "0 <= i2 & i2 < sa..csize" */
1602     sa.set(i2, v2);
1603
1604     sb.set(i2, v2);
1605     Object r1b = sb.get(i1);
1606
1607     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
1608 }
1609
1610 static void get_set_pre_c_48(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)

```



```

1608  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1609          sa..contents = sb..contents & sa..csize = sb..csize"
1610  modifies "sa..contents", "sb..contents"
1611  ensures "True" */
1612  {
1613      /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 <
          sa..csize) | i1 > i2)" */
1614      /*: assume "0 <= i1 & i1 < sa..csize" */
1615      Object r1a = sa.get(i1);
1616      /*: assume "0 <= i2 & i2 < sa..csize" */
1617      sa.set(i2, v2);
1618
1619      /*: assume "0 <= i2 & i2 < sb..csize" */
1620      sb.set(i2, v2);
1621      /*: assume "0 <= i1 & i1 < sb..csize" */
1622      Object r1b = sb.get(i1);
1623
1624      /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
          */
1625  }
1626
1627  static void get_set_between_s_49(ArrayList sa, ArrayList sb, int i1, int i2, Object
          v2)
1628  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1629          sa..contents = sb..contents & sa..csize = sb..csize"
1630  modifies "sa..contents", "sb..contents"
1631  ensures "True" */
1632  {
1633      /*: assume "0 <= i1 & i1 < sa..csize" */
1634      Object r1a = sa.get(i1);
1635      /*: assume "i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2" */
1636      /*: assume "0 <= i2 & i2 < sa..csize" */
1637      sa.set(i2, v2);
1638
1639      sb.set(i2, v2);
1640      Object r1b = sb.get(i1);
1641
1642      /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
1643  }
1644
1645  static void get_set_between_c_49(ArrayList sa, ArrayList sb, int i1, int i2, Object
          v2)
1646  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1647          sa..contents = sb..contents & sa..csize = sb..csize"
1648  modifies "sa..contents", "sb..contents"
1649  ensures "True" */
1650  {
1651      /*: assume "0 <= i1 & i1 < sa..csize" */
1652      Object r1a = sa.get(i1);
1653      /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2)" */
1654      /*: assume "0 <= i2 & i2 < sa..csize" */
1655      sa.set(i2, v2);
1656
1657      /*: assume "0 <= i2 & i2 < sb..csize" */
1658      sb.set(i2, v2);
1659      /*: assume "0 <= i1 & i1 < sb..csize" */
1660      Object r1b = sb.get(i1);
1661
1662      /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
          */
1663  }
1664
1665  static void get_set_post_s_50(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
1666  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1667          sa..contents = sb..contents & sa..csize = sb..csize"

```

```

1668     modifies "sa..contents", "sb..contents"
1669     ensures "True" */
1670 {
1671     /*: assume "0 <= i1 & i1 < sa..csize" */
1672     Object r1a = sa.get(i1);
1673     /*: assume "0 <= i2 & i2 < sa..csize" */
1674     sa.set(i2, v2);
1675     /*: assume "i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2" */
1676
1677     sb.set(i2, v2);
1678     Object r1b = sb.get(i1);
1679
1680     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
1681 }
1682
1683 static void get_set_post_c_50(ArrayList sa, ArrayList sb, int i1, int i2, Object v2)
1684 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1685     sa..contents = sb..contents & sa..csize = sb..csize"
1686     modifies "sa..contents", "sb..contents"
1687     ensures "True" */
1688 {
1689     /*: assume "0 <= i1 & i1 < sa..csize" */
1690     Object r1a = sa.get(i1);
1691     /*: assume "0 <= i2 & i2 < sa..csize" */
1692     sa.set(i2, v2);
1693     /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v2) | i1 > i2)" */
1694
1695     /*: assume "0 <= i2 & i2 < sb..csize" */
1696     sb.set(i2, v2);
1697     /*: assume "0 <= i1 & i1 < sb..csize" */
1698     Object r1b = sb.get(i1);
1699
1700     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
1701         */
1702 }
1703
1704 static void get_size_pre_s_51(ArrayList sa, ArrayList sb, int i1)
1705 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1706     sa..contents = sb..contents & sa..csize = sb..csize"
1707     ensures "True" */
1708 {
1709     /*: assume "True" */
1710     /*: assume "0 <= i1 & i1 < sa..csize" */
1711     Object r1a = sa.get(i1);
1712     int r2a = sa.size();
1713
1714     int r2b = sb.size();
1715     Object r1b = sb.get(i1);
1716
1717     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1718         sb..csize" */
1719 }
1720
1721 static void get_size_pre_c_51(ArrayList sa, ArrayList sb, int i1)
1722 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1723     sa..contents = sb..contents & sa..csize = sb..csize"
1724     ensures "True" */
1725 {
1726     /*: assume "~(True)" */
1727     /*: assume "0 <= i1 & i1 < sa..csize" */
1728     Object r1a = sa.get(i1);
1729     int r2a = sa.size();
1730
1731     int r2b = sb.size();
1732     /*: assume "0 <= i1 & i1 < sb..csize" */

```

```

1731     Object r1b = sb.get(i1);
1732
1733     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
1734 }
1735
1736 static void get_size_between_s_52(ArrayList sa, ArrayList sb, int i1)
1737 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1738     sa..contents = sb..contents & sa..csize = sb..csize"
1739     ensures "True" */
1740 {
1741     /*: assume "0 <= i1 & i1 < sa..csize" */
1742     Object r1a = sa.get(i1);
1743     /*: assume "True" */
1744     int r2a = sa.size();
1745
1746     int r2b = sb.size();
1747     Object r1b = sb.get(i1);
1748
1749     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
1750 }
1751
1752 static void get_size_between_c_52(ArrayList sa, ArrayList sb, int i1)
1753 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1754     sa..contents = sb..contents & sa..csize = sb..csize"
1755     ensures "True" */
1756 {
1757     /*: assume "0 <= i1 & i1 < sa..csize" */
1758     Object r1a = sa.get(i1);
1759     /*: assume "~(True)" */
1760     int r2a = sa.size();
1761
1762     int r2b = sb.size();
1763     /*: assume "0 <= i1 & i1 < sb..csize" */
1764     Object r1b = sb.get(i1);
1765
1766     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
1767 }
1768
1769 static void get_size_post_s_53(ArrayList sa, ArrayList sb, int i1)
1770 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1771     sa..contents = sb..contents & sa..csize = sb..csize"
1772     ensures "True" */
1773 {
1774     /*: assume "0 <= i1 & i1 < sa..csize" */
1775     Object r1a = sa.get(i1);
1776     int r2a = sa.size();
1777     /*: assume "True" */
1778
1779     int r2b = sb.size();
1780     Object r1b = sb.get(i1);
1781
1782     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
1783 }
1784
1785 static void get_size_post_c_53(ArrayList sa, ArrayList sb, int i1)
1786 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1787     sa..contents = sb..contents & sa..csize = sb..csize"
1788     ensures "True" */
1789 {
1790     /*: assume "0 <= i1 & i1 < sa..csize" */
1791     Object r1a = sa.get(i1);

```

```

1792     int r2a = sa.size();
1793     /*: assume "~(True)" */
1794
1795     int r2b = sb.size();
1796     /*: assume "0 <= i1 & i1 < sb..csize" */
1797     Object r1b = sb.get(i1);
1798
1799     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1800         sb..csize)" */
1801 }
1802
1803 static void indexOf_add_at_pre_s_54(ArrayList sa, ArrayList sb, Object v1, int i2,
1804     Object v2)
1805 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1806     sa..contents = sb..contents & sa..csize = sb..csize"
1807 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
1808     "sb..msize"
1809 ensures "True" */
1810 {
1811     /*: assume "~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
1812         v2) | (EX i. (i, v1) : sa..contents & 0 <= i & i < i2) | ~(EX i. (i, v1) :
1813         sa..contents & 0 <= i & i < i2) & (i2, v1) : sa..contents & 0 <= i2 & i2 <
1814         sa..csize & v1 = v2)" */
1815     int r1a = sa.indexOf(v1);
1816     /*: assume "0 <= i2 & i2 <= sa..csize" */
1817     sa.add_at(i2, v2);
1818
1819     sb.add_at(i2, v2);
1820     int r1b = sb.indexOf(v1);
1821
1822     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
1823 }
1824
1825 static void indexOf_add_at_pre_c_54(ArrayList sa, ArrayList sb, Object v1, int i2,
1826     Object v2)
1827 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1828     sa..contents = sb..contents & sa..csize = sb..csize"
1829 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
1830     "sb..msize"
1831 ensures "True" */
1832 {
1833     /*: assume "~((~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
1834         v2) | (EX i. (i, v1) : sa..contents & 0 <= i & i < i2) | ~(EX i. (i, v1) :
1835         sa..contents & 0 <= i & i < i2) & (i2, v1) : sa..contents & 0 <= i2 & i2 <
1836         sa..csize & v1 = v2))" */
1837     int r1a = sa.indexOf(v1);
1838     /*: assume "0 <= i2 & i2 <= sa..csize" */
1839     sa.add_at(i2, v2);
1840
1841     /*: assume "0 <= i2 & i2 <= sb..csize" */
1842     sb.add_at(i2, v2);
1843     int r1b = sb.indexOf(v1);
1844
1845     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
1846         */
1847 }
1848
1849 static void indexOf_add_at_between_s_55(ArrayList sa, ArrayList sb, Object v1, int
1850     i2, Object v2)
1851 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1852     sa..contents = sb..contents & sa..csize = sb..csize"
1853 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
1854     "sb..msize"
1855 ensures "True" */
1856 {

```

```

1843     int r1a = sa.indexOf(v1);
1844     /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 =
1845         v2)" */
1846     /*: assume "0 <= i2 & i2 <= sa..csize" */
1847     sa.add_at(i2, v2);
1848
1849     sb.add_at(i2, v2);
1850     int r1b = sb.indexOf(v1);
1851
1852     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
1853 }
1854
1855 static void indexOf_add_at_between_c_55(ArrayList sa, ArrayList sb, Object v1, int
1856     i2, Object v2)
1857 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1858     sa..contents = sb..contents & sa..csize = sb..csize"
1859     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
1860     "sb..msize"
1861     ensures "True" */
1862 {
1863     int r1a = sa.indexOf(v1);
1864     /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 =
1865         v2))" */
1866     /*: assume "0 <= i2 & i2 <= sa..csize" */
1867     sa.add_at(i2, v2);
1868
1869     /*: assume "0 <= i2 & i2 <= sb..csize" */
1870     sb.add_at(i2, v2);
1871     int r1b = sb.indexOf(v1);
1872
1873     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
1874         */
1875 }
1876
1877 static void indexOf_add_at_post_s_56(ArrayList sa, ArrayList sb, Object v1, int i2,
1878     Object v2)
1879 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1880     sa..contents = sb..contents & sa..csize = sb..csize"
1881     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
1882     "sb..msize"
1883     ensures "True" */
1884 {
1885     int r1a = sa.indexOf(v1);
1886     /*: assume "0 <= i2 & i2 <= sa..csize" */
1887     sa.add_at(i2, v2);
1888     /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 =
1889         v2)" */
1890
1891     sb.add_at(i2, v2);
1892     int r1b = sb.indexOf(v1);
1893
1894     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
1895 }
1896
1897 static void indexOf_add_at_post_c_56(ArrayList sa, ArrayList sb, Object v1, int i2,
1898     Object v2)
1899 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1900     sa..contents = sb..contents & sa..csize = sb..csize"
1901     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
1902     "sb..msize"
1903     ensures "True" */
1904 {
1905     int r1a = sa.indexOf(v1);
1906     /*: assume "0 <= i2 & i2 <= sa..csize" */
1907     sa.add_at(i2, v2);

```

```

1898     /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 =
1899         v2))" */
1900
1901     /*: assume "0 <= i2 & i2 <= sb..csize" */
1902     sb.add_at(i2, v2);
1903     int r1b = sb.indexOf(v1);
1904
1905     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
1906         */
1907 }
1908
1909 static void indexOf_get_pre_s_57(ArrayList sa, ArrayList sb, Object v1, int i2)
1910 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1911     sa..contents = sb..contents & sa..csize = sb..csize"
1912     ensures "True" */
1913 {
1914     /*: assume "True" */
1915     int r1a = sa.indexOf(v1);
1916     /*: assume "0 <= i2 & i2 < sa..csize" */
1917     Object r2a = sa.get(i2);
1918
1919     Object r2b = sb.get(i2);
1920     int r1b = sb.indexOf(v1);
1921
1922     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1923         sb..csize" */
1924 }
1925
1926 static void indexOf_get_pre_c_57(ArrayList sa, ArrayList sb, Object v1, int i2)
1927 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1928     sa..contents = sb..contents & sa..csize = sb..csize"
1929     ensures "True" */
1930 {
1931     /*: assume "~(True)" */
1932     int r1a = sa.indexOf(v1);
1933     /*: assume "0 <= i2 & i2 < sa..csize" */
1934     Object r2a = sa.get(i2);
1935
1936     /*: assume "0 <= i2 & i2 < sb..csize" */
1937     Object r2b = sb.get(i2);
1938     int r1b = sb.indexOf(v1);
1939
1940     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1941         sb..csize)" */
1942 }
1943
1944 static void indexOf_get_between_s_58(ArrayList sa, ArrayList sb, Object v1, int i2)
1945 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1946     sa..contents = sb..contents & sa..csize = sb..csize"
1947     ensures "True" */
1948 {
1949     int r1a = sa.indexOf(v1);
1950     /*: assume "True" */
1951     /*: assume "0 <= i2 & i2 < sa..csize" */
1952     Object r2a = sa.get(i2);
1953
1954     Object r2b = sb.get(i2);
1955     int r1b = sb.indexOf(v1);
1956
1957     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
1958         sb..csize" */
1959 }
1960
1961 static void indexOf_get_between_c_58(ArrayList sa, ArrayList sb, Object v1, int i2)
1962 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &

```

```

1958         sa..contents = sb..contents & sa..csize = sb..csize"
1959     ensures "True" */
1960 {
1961     int r1a = sa.indexOf(v1);
1962     /*: assume "~(True)" */
1963     /*: assume "0 <= i2 & i2 < sa..csize" */
1964     Object r2a = sa.get(i2);
1965
1966     /*: assume "0 <= i2 & i2 < sb..csize" */
1967     Object r2b = sb.get(i2);
1968     int r1b = sb.indexOf(v1);
1969
1970     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
1971 }
1972
1973 static void indexOf_get_post_s_59(ArrayList sa, ArrayList sb, Object v1, int i2)
1974 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1975     sa..contents = sb..contents & sa..csize = sb..csize"
1976     ensures "True" */
1977 {
1978     int r1a = sa.indexOf(v1);
1979     /*: assume "0 <= i2 & i2 < sa..csize" */
1980     Object r2a = sa.get(i2);
1981     /*: assume "True" */
1982
1983     Object r2b = sb.get(i2);
1984     int r1b = sb.indexOf(v1);
1985
1986     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
1987 }
1988
1989 static void indexOf_get_post_c_59(ArrayList sa, ArrayList sb, Object v1, int i2)
1990 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1991     sa..contents = sb..contents & sa..csize = sb..csize"
1992     ensures "True" */
1993 {
1994     int r1a = sa.indexOf(v1);
1995     /*: assume "0 <= i2 & i2 < sa..csize" */
1996     Object r2a = sa.get(i2);
1997     /*: assume "~(True)" */
1998
1999     /*: assume "0 <= i2 & i2 < sb..csize" */
2000     Object r2b = sb.get(i2);
2001     int r1b = sb.indexOf(v1);
2002
2003     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
2004 }
2005
2006 static void indexOf_indexOf_pre_s_60(ArrayList sa, ArrayList sb, Object v1, Object
    v2)
2007 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2008     sa..contents = sb..contents & sa..csize = sb..csize"
2009     ensures "True" */
2010 {
2011     /*: assume "True" */
2012     int r1a = sa.indexOf(v1);
2013     int r2a = sa.indexOf(v2);
2014
2015     int r2b = sb.indexOf(v2);
2016     int r1b = sb.indexOf(v1);
2017

```

```

2018     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2019         sb..csize" */
2019 }
2020
2021 static void indexOf_indexOf_pre_c_60(ArrayList sa, ArrayList sb, Object v1, Object
2022     v2)
2023 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2024     sa..contents = sb..contents & sa..csize = sb..csize"
2025     ensures "True" */
2025 {
2026     /*: assume "~(True)" */
2027     int r1a = sa.indexOf(v1);
2028     int r2a = sa.indexOf(v2);
2029
2030     int r2b = sb.indexOf(v2);
2031     int r1b = sb.indexOf(v1);
2032
2033     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2034         sb..csize)" */
2034 }
2035
2036 static void indexOf_indexOf_between_s_61(ArrayList sa, ArrayList sb, Object v1,
2037     Object v2)
2038 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2039     sa..contents = sb..contents & sa..csize = sb..csize"
2040     ensures "True" */
2040 {
2041     int r1a = sa.indexOf(v1);
2042     /*: assume "True" */
2043     int r2a = sa.indexOf(v2);
2044
2045     int r2b = sb.indexOf(v2);
2046     int r1b = sb.indexOf(v1);
2047
2048     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2049         sb..csize" */
2049 }
2050
2051 static void indexOf_indexOf_between_c_61(ArrayList sa, ArrayList sb, Object v1,
2052     Object v2)
2053 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2054     sa..contents = sb..contents & sa..csize = sb..csize"
2055     ensures "True" */
2055 {
2056     int r1a = sa.indexOf(v1);
2057     /*: assume "~(True)" */
2058     int r2a = sa.indexOf(v2);
2059
2060     int r2b = sb.indexOf(v2);
2061     int r1b = sb.indexOf(v1);
2062
2063     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2064         sb..csize)" */
2064 }
2065
2066 static void indexOf_indexOf_post_s_62(ArrayList sa, ArrayList sb, Object v1, Object
2067     v2)
2068 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2069     sa..contents = sb..contents & sa..csize = sb..csize"
2070     ensures "True" */
2070 {
2071     int r1a = sa.indexOf(v1);
2072     int r2a = sa.indexOf(v2);
2073     /*: assume "True" */
2074

```



```

2075     int r2b = sb.indexOf(v2);
2076     int r1b = sb.indexOf(v1);
2077
2078     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
2079 }
2080
2081 static void indexOf_indexOf_post_c_62(ArrayList sa, ArrayList sb, Object v1, Object
    v2)
2082 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2083     sa..contents = sb..contents & sa..csize = sb..csize"
2084     ensures "True" */
2085 {
2086     int r1a = sa.indexOf(v1);
2087     int r2a = sa.indexOf(v2);
2088     /*: assume "~(True)" */
2089
2090     int r2b = sb.indexOf(v2);
2091     int r1b = sb.indexOf(v1);
2092
2093     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
2094 }
2095
2096 static void indexOf_lastIndexOf_pre_s_63(ArrayList sa, ArrayList sb, Object v1,
    Object v2)
2097 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2098     sa..contents = sb..contents & sa..csize = sb..csize"
2099     ensures "True" */
2100 {
2101     /*: assume "True" */
2102     int r1a = sa.indexOf(v1);
2103     int r2a = sa.lastIndexOf(v2);
2104
2105     int r2b = sb.lastIndexOf(v2);
2106     int r1b = sb.indexOf(v1);
2107
2108     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
2109 }
2110
2111 static void indexOf_lastIndexOf_pre_c_63(ArrayList sa, ArrayList sb, Object v1,
    Object v2)
2112 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2113     sa..contents = sb..contents & sa..csize = sb..csize"
2114     ensures "True" */
2115 {
2116     /*: assume "~(True)" */
2117     int r1a = sa.indexOf(v1);
2118     int r2a = sa.lastIndexOf(v2);
2119
2120     int r2b = sb.lastIndexOf(v2);
2121     int r1b = sb.indexOf(v1);
2122
2123     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
2124 }
2125
2126 static void indexOf_lastIndexOf_between_s_64(ArrayList sa, ArrayList sb, Object v1,
    Object v2)
2127 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2128     sa..contents = sb..contents & sa..csize = sb..csize"
2129     ensures "True" */
2130 {
2131     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 &
        sa..contents = sb..contents & sa..csize = sb..csize"
2143     ensures "True" */
2144 {
2145     int r1a = sa.indexOf(v1);
2146     /*: assume "~(True)" */
2147     int r2a = sa.lastIndexOf(v2);
2148
2149     int r2b = sb.lastIndexOf(v2);
2150     int r1b = sb.indexOf(v1);
2151
2152     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
2153 }
2154
2155 static void indexOf_lastIndexOf_post_s_65(ArrayList sa, ArrayList sb, Object v1,
        Object v2)
2156 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
2157     ensures "True" */
2158 {
2159     int r1a = sa.indexOf(v1);
2160     int r2a = sa.lastIndexOf(v2);
2161     /*: assume "True" */
2162
2163     int r2b = sb.lastIndexOf(v2);
2164     int r1b = sb.indexOf(v1);
2165
2166     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
2167 }
2168
2169 static void indexOf_lastIndexOf_post_c_65(ArrayList sa, ArrayList sb, Object v1,
        Object v2)
2170 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
2171     ensures "True" */
2172 {
2173     int r1a = sa.indexOf(v1);
2174     int r2a = sa.lastIndexOf(v2);
2175     /*: assume "~(True)" */
2176
2177     int r2b = sb.lastIndexOf(v2);
2178     int r1b = sb.indexOf(v1);
2179
2180     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
2181 }
2182
2183 static void indexOf_remove_at_post_c_68(ArrayList sa, ArrayList sb, Object v1, int
        i2)
2184 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
2185
2186
2187
2188

```

```

2189     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2190     ensures "True" */
2191 {
2192     int r1a = sa.indexOf(v1);
2193     /*: assume "0 <= i2 & i2 < sa..csize" */
2194     Object r2a = sa.remove_at(i2);
2195     /*: assume "~(r1a < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize &
        (i2, v1) : sa..contents & 0 <= i2 & i2 < sa..csize))" */
2196
2197     /*: assume "0 <= i2 & i2 < sb..csize" */
2198     Object r2b = sb.remove_at(i2);
2199     int r1b = sb.indexOf(v1);
2200
2201     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
2202 }
2203
2204 static void indexOf_set_pre_s_72(ArrayList sa, ArrayList sb, Object v1, int i2,
    Object v2)
2205 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2206     sa..contents = sb..contents & sa..csize = sb..csize"
2207     modifies "sa..contents", "sb..contents"
2208     ensures "True" */
2209 {
2210     /*: 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)" */
2211     int r1a = sa.indexOf(v1);
2212     /*: assume "0 <= i2 & i2 < sa..csize" */
2213     Object r2a = sa.set(i2, v2);
2214
2215     Object r2b = sb.set(i2, v2);
2216     int r1b = sb.indexOf(v1);
2217
2218     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
2219 }
2220
2221 static void indexOf_set_pre_c_72(ArrayList sa, ArrayList sb, Object v1, int i2,
    Object v2)
2222 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2223     sa..contents = sb..contents & sa..csize = sb..csize"
2224     modifies "sa..contents", "sb..contents"
2225     ensures "True" */
2226 {
2227     /*: 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)" */
2228     int r1a = sa.indexOf(v1);
2229     /*: assume "0 <= i2 & i2 < sa..csize" */
2230     Object r2a = sa.set(i2, v2);
2231
2232     /*: assume "0 <= i2 & i2 < sb..csize" */
2233     Object r2b = sb.set(i2, v2);
2234     int r1b = sb.indexOf(v1);
2235
2236     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
2237 }
2238

```

```

2239 static void indexOf_set_between_s_73(ArrayList sa, ArrayList sb, Object v1, int i2,
2240 Object v2)
2241 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2242 sa..contents = sb..contents & sa..csize = sb..csize"
2243 modifies "sa..contents", "sb..contents"
2244 ensures "True" */
2245 {
2246 int r1a = sa.indexOf(v1);
2247 /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 = v2)
2248 | (r1a > i2 & v1 ~= v2)" */
2249 /*: assume "0 <= i2 & i2 < sa..csize" */
2250 Object r2a = sa.set(i2, v2);
2251
2252 Object r2b = sb.set(i2, v2);
2253 int r1b = sb.indexOf(v1);
2254
2255 /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2256 sb..csize" */
2257 }
2258
2259 static void indexOf_set_between_c_73(ArrayList sa, ArrayList sb, Object v1, int i2,
2260 Object v2)
2261 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2262 sa..contents = sb..contents & sa..csize = sb..csize"
2263 modifies "sa..contents", "sb..contents"
2264 ensures "True" */
2265 {
2266 int r1a = sa.indexOf(v1);
2267 /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 =
2268 v2) | (r1a > i2 & v1 ~= v2))" */
2269 /*: assume "0 <= i2 & i2 < sa..csize" */
2270 Object r2a = sa.set(i2, v2);
2271
2272 /*: assume "0 <= i2 & i2 < sb..csize" */
2273 Object r2b = sb.set(i2, v2);
2274 int r1b = sb.indexOf(v1);
2275
2276 /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2277 sb..csize)" */
2278 }
2279
2280 static void indexOf_set_post_s_74(ArrayList sa, ArrayList sb, Object v1, int i2,
2281 Object v2)
2282 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2283 sa..contents = sb..contents & sa..csize = sb..csize"
2284 modifies "sa..contents", "sb..contents"
2285 ensures "True" */
2286 {
2287 int r1a = sa.indexOf(v1);
2288 /*: assume "0 <= i2 & i2 < sa..csize" */
2289 Object r2a = sa.set(i2, v2);
2290 /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 = v2)
2291 | (r1a > i2 & v1 ~= v2)" */
2292
2293 Object r2b = sb.set(i2, v2);
2294 int r1b = sb.indexOf(v1);
2295
2296 /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2297 sb..csize" */
2298 }
2299
2300 static void indexOf_set_post_c_74(ArrayList sa, ArrayList sb, Object v1, int i2,
2301 Object v2)
2302 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2303 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 =
2301         v2) | (r1a > i2 & v1 ~= v2))" */
2302
2303     /*: assume "0 <= i2 & i2 < sb..csize" */
2304     Object r2b = sb.set(i2, v2);
2305     int r1b = sb.indexOf(v1);
2306
2307     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2308         sb..csize)" */
2309 }
2310
2311 static void indexOf_set_pre_s_75(ArrayList sa, ArrayList sb, Object v1, int i2,
2312     Object v2)
2313 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2314     sa..contents = sb..contents & sa..csize = sb..csize"
2315     modifies "sa..contents", "sb..contents"
2316     ensures "True" */
2317 {
2318     /*: assume "~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
2319         v2) | (EX i. (i, v1) : sa..contents & 0 <= i & i < i2) | (~(EX i. (i, v1) :
2320         sa..contents & 0 <= i & i < i2) & (i2, v1) : sa..contents & v1 = v2) | (~(EX
2321         i. (i, v1) : sa..contents & 0 <= i & i <= i2) & (EX i. (i, v1) :
2322         sa..contents & i2 < i & i < sa..csize) & v1 ~= v2)" */
2323     int r1a = sa.indexOf(v1);
2324     /*: assume "0 <= i2 & i2 < sa..csize" */
2325     sa.set(i2, v2);
2326
2327     sb.set(i2, v2);
2328     int r1b = sb.indexOf(v1);
2329
2330     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
2331 }
2332
2333 static void indexOf_set_pre_c_75(ArrayList sa, ArrayList sb, Object v1, int i2,
2334     Object v2)
2335 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2336     sa..contents = sb..contents & sa..csize = sb..csize"
2337     modifies "sa..contents", "sb..contents"
2338     ensures "True" */
2339 {
2340     /*: assume "~((~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
2341         v2) | (EX i. (i, v1) : sa..contents & 0 <= i & i < i2) | (~(EX i. (i, v1) :
2342         sa..contents & 0 <= i & i < i2) & (i2, v1) : sa..contents & v1 = v2) | (~(EX
2343         i. (i, v1) : sa..contents & 0 <= i & i <= i2) & (EX i. (i, v1) :
2344         sa..contents & i2 < i & i < sa..csize) & v1 ~= v2))" */
2345     int r1a = sa.indexOf(v1);
2346     /*: assume "0 <= i2 & i2 < sa..csize" */
2347     sa.set(i2, v2);
2348
2349     /*: assume "0 <= i2 & i2 < sb..csize" */
2350     sb.set(i2, v2);
2351     int r1b = sb.indexOf(v1);
2352
2353     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
2354         */
2355 }
2356
2357 static void indexOf_set_between_s_76(ArrayList sa, ArrayList sb, Object v1, int i2,
2358     Object v2)

```

```

2345  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2346           sa..contents = sb..contents & sa..csize = sb..csize"
2347  modifies "sa..contents", "sb..contents"
2348  ensures "True" */
2349  {
2350      int r1a = sa.indexOf(v1);
2351      /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 = v2)
           | (r1a > i2 & v1 ~= v2)" */
2352      /*: assume "0 <= i2 & i2 < sa..csize" */
2353      sa.set(i2, v2);
2354
2355      sb.set(i2, v2);
2356      int r1b = sb.indexOf(v1);
2357
2358      /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
2359  }
2360
2361  static void indexOf_set_between_c_76(ArrayList sa, ArrayList sb, Object v1, int i2,
           Object v2)
2362  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
           sa..contents = sb..contents & sa..csize = sb..csize"
2363  modifies "sa..contents", "sb..contents"
2364  ensures "True" */
2365  {
2366      int r1a = sa.indexOf(v1);
2367      /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 =
           v2) | (r1a > i2 & v1 ~= v2))" */
2368      /*: assume "0 <= i2 & i2 < sa..csize" */
2369      sa.set(i2, v2);
2370
2371
2372      /*: assume "0 <= i2 & i2 < sb..csize" */
2373      sb.set(i2, v2);
2374      int r1b = sb.indexOf(v1);
2375
2376      /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
           */
2377  }
2378
2379  static void indexOf_set_post_s_77(ArrayList sa, ArrayList sb, Object v1, int i2,
           Object v2)
2380  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
           sa..contents = sb..contents & sa..csize = sb..csize"
2381  modifies "sa..contents", "sb..contents"
2382  ensures "True" */
2383  {
2384      int r1a = sa.indexOf(v1);
2385      /*: assume "0 <= i2 & i2 < sa..csize" */
2386      sa.set(i2, v2);
2387      /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 = v2)
           | (r1a > i2 & v1 ~= v2)" */
2388
2389      sb.set(i2, v2);
2390      int r1b = sb.indexOf(v1);
2391
2392      /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
2393  }
2394
2395  static void indexOf_set_post_c_77(ArrayList sa, ArrayList sb, Object v1, int i2,
           Object v2)
2396  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
           sa..contents = sb..contents & sa..csize = sb..csize"
2397  modifies "sa..contents", "sb..contents"
2398  ensures "True" */
2399  {
2400      int r1a = sa.indexOf(v1);
2401
2402

```

```

2403     /*: assume "0 <= i2 & i2 < sa..csize" */
2404     sa.set(i2, v2);
2405     /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2) | (r1a = i2 & v1 =
2406         v2) | (r1a > i2 & v1 ~= v2))" */
2407
2408     /*: assume "0 <= i2 & i2 < sb..csize" */
2409     sb.set(i2, v2);
2410     int r1b = sb.indexOf(v1);
2411
2412     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
2413         */
2414 }
2415
2416 static void indexOf_size_pre_s_78(ArrayList sa, ArrayList sb, Object v1)
2417 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2418     sa..contents = sb..contents & sa..csize = sb..csize"
2419     ensures "True" */
2420 {
2421     /*: assume "True" */
2422     int r1a = sa.indexOf(v1);
2423     int r2a = sa.size();
2424
2425     int r2b = sb.size();
2426     int r1b = sb.indexOf(v1);
2427
2428     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2429         sb..csize" */
2430 }
2431
2432 static void indexOf_size_pre_c_78(ArrayList sa, ArrayList sb, Object v1)
2433 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2434     sa..contents = sb..contents & sa..csize = sb..csize"
2435     ensures "True" */
2436 {
2437     /*: assume "~(True)" */
2438     int r1a = sa.indexOf(v1);
2439     int r2a = sa.size();
2440
2441     int r2b = sb.size();
2442     int r1b = sb.indexOf(v1);
2443
2444     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2445         sb..csize)" */
2446 }
2447
2448 static void indexOf_size_between_s_79(ArrayList sa, ArrayList sb, Object v1)
2449 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2450     sa..contents = sb..contents & sa..csize = sb..csize"
2451     ensures "True" */
2452 {
2453     int r1a = sa.indexOf(v1);
2454     /*: assume "True" */
2455     int r2a = sa.size();
2456
2457     int r2b = sb.size();
2458     int r1b = sb.indexOf(v1);
2459
2460     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2461         sb..csize" */
2462 }
2463
2464 static void indexOf_size_between_c_79(ArrayList sa, ArrayList sb, Object v1)
2465 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2466     sa..contents = sb..contents & sa..csize = sb..csize"
2467     ensures "True" */

```

```

2463 {
2464     int r1a = sa.indexOf(v1);
2465     /*: assume "~(True)" */
2466     int r2a = sa.size();
2467
2468     int r2b = sb.size();
2469     int r1b = sb.indexOf(v1);
2470
2471     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
2472 }
2473
2474 static void indexOf_size_post_s_80(ArrayList sa, ArrayList sb, Object v1)
2475 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
2476     ensures "True" */
2477 {
2478     int r1a = sa.indexOf(v1);
2479     int r2a = sa.size();
2480     /*: assume "True" */
2481
2482     int r2b = sb.size();
2483     int r1b = sb.indexOf(v1);
2484
2485     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
2486 }
2487
2488 static void indexOf_size_post_c_80(ArrayList sa, ArrayList sb, Object v1)
2489 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
2490     ensures "True" */
2491 {
2492     int r1a = sa.indexOf(v1);
2493     int r2a = sa.size();
2494     /*: assume "~(True)" */
2495
2496     int r2b = sb.size();
2497     int r1b = sb.indexOf(v1);
2498
2499     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
2500 }
2501
2502 static void lastIndexOf_add_at_pre_s_81(ArrayList sa, ArrayList sb, Object v1, int
        i2, Object v2)
2503 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
2504     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
        "sb..msize"
2505     ensures "True" */
2506 {
2507     /*: 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)" */
2508     int r1a = sa.lastIndexOf(v1);
2509     /*: assume "0 <= i2 & i2 <= sa..csize" */
2510     sa.add_at(i2, v2);
2511
2512     sb.add_at(i2, v2);
2513     int r1b = sb.lastIndexOf(v1);
2514
2515     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
2516 }
2517
2518
2519
2520

```



```

2521 static void lastIndexOf_add_at_between_s_82(ArrayList sa, ArrayList sb, Object v1,
2522 int i2, Object v2)
2523 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2524 sa..contents = sb..contents & sa..csize = sb..csize"
2525 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
2526 "sb..msize"
2527 ensures "True" */
2528 {
2529 int r1a = sa.lastIndexOf(v1);
2530 /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2)" */
2531 /*: assume "0 <= i2 & i2 <= sa..csize" */
2532 sa.add_at(i2, v2);
2533
2534 sb.add_at(i2, v2);
2535 int r1b = sb.lastIndexOf(v1);
2536
2537 /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
2538 }
2539
2540 static void lastIndexOf_add_at_post_s_83(ArrayList sa, ArrayList sb, Object v1, int
2541 i2, Object v2)
2542 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2543 sa..contents = sb..contents & sa..csize = sb..csize"
2544 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
2545 "sb..msize"
2546 ensures "True" */
2547 {
2548 int r1a = sa.lastIndexOf(v1);
2549 /*: assume "0 <= i2 & i2 <= sa..csize" */
2550 sa.add_at(i2, v2);
2551 /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2)" */
2552
2553 sb.add_at(i2, v2);
2554 int r1b = sb.lastIndexOf(v1);
2555
2556 /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
2557 }
2558
2559 static void lastIndexOf_get_pre_s_84(ArrayList sa, ArrayList sb, Object v1, int i2)
2560 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2561 sa..contents = sb..contents & sa..csize = sb..csize"
2562 ensures "True" */
2563 {
2564 /*: assume "True" */
2565 int r1a = sa.lastIndexOf(v1);
2566 /*: assume "0 <= i2 & i2 < sa..csize" */
2567 Object r2a = sa.get(i2);
2568
2569 Object r2b = sb.get(i2);
2570 int r1b = sb.lastIndexOf(v1);
2571
2572 /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2573 sb..csize" */
2574 }
2575
2576 static void lastIndexOf_get_pre_c_84(ArrayList sa, ArrayList sb, Object v1, int i2)
2577 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2578 sa..contents = sb..contents & sa..csize = sb..csize"
2579 ensures "True" */
2580 {
2581 /*: assume "~(True)" */
2582 int r1a = sa.lastIndexOf(v1);
2583 /*: assume "0 <= i2 & i2 < sa..csize" */
2584 Object r2a = sa.get(i2);

```

```

2581     /*: assume "0 <= i2 & i2 < sb..csize" */
2582     Object r2b = sb.get(i2);
2583     int r1b = sb.lastIndexOf(v1);
2584
2585     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
2586 }
2587
2588 static void lastIndexOf_get_between_s_85(ArrayList sa, ArrayList sb, Object v1, int
        i2)
2589 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
2590     ensures "True" */
2591 {
2592     int r1a = sa.lastIndexOf(v1);
2593     /*: assume "True" */
2594     /*: assume "0 <= i2 & i2 < sa..csize" */
2595     Object r2a = sa.get(i2);
2596
2597     Object r2b = sb.get(i2);
2598     int r1b = sb.lastIndexOf(v1);
2599
2600     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
2601 }
2602
2603 static void lastIndexOf_get_between_c_85(ArrayList sa, ArrayList sb, Object v1, int
        i2)
2604 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
2605     ensures "True" */
2606 {
2607     int r1a = sa.lastIndexOf(v1);
2608     /*: assume "(True)" */
2609     /*: assume "0 <= i2 & i2 < sa..csize" */
2610     Object r2a = sa.get(i2);
2611
2612     /*: assume "0 <= i2 & i2 < sb..csize" */
2613     Object r2b = sb.get(i2);
2614     int r1b = sb.lastIndexOf(v1);
2615
2616     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
2617 }
2618
2619 static void lastIndexOf_get_post_s_86(ArrayList sa, ArrayList sb, Object v1, int i2)
2620 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
2621     ensures "True" */
2622 {
2623     int r1a = sa.lastIndexOf(v1);
2624     /*: assume "0 <= i2 & i2 < sa..csize" */
2625     Object r2a = sa.get(i2);
2626     /*: assume "True" */
2627
2628     Object r2b = sb.get(i2);
2629     int r1b = sb.lastIndexOf(v1);
2630
2631     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
2632 }
2633
2634 static void lastIndexOf_get_post_c_86(ArrayList sa, ArrayList sb, Object v1, int i2)
2635 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
2636     ensures "True" */
2637 {
2638     int r1a = sa.lastIndexOf(v1);
2639     /*: assume "(True)" */
2640     /*: assume "0 <= i2 & i2 < sa..csize" */
2641     Object r2a = sa.get(i2);
2642
2643     /*: assume "0 <= i2 & i2 < sb..csize" */
2644     Object r2b = sb.get(i2);
2645     int r1b = sb.lastIndexOf(v1);
2646
2647     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
2648 }

```

```

2640     ensures "True" */
2641 {
2642     int r1a = sa.lastIndexOf(v1);
2643     /*: assume "0 <= i2 & i2 < sa..csize" */
2644     Object r2a = sa.get(i2);
2645     /*: assume "~(True)" */
2646
2647     /*: assume "0 <= i2 & i2 < sb..csize" */
2648     Object r2b = sb.get(i2);
2649     int r1b = sb.lastIndexOf(v1);
2650
2651     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2652         sb..csize)" */
2653 }
2654
2655 static void lastIndexOf_indexOf_pre_s_87(ArrayList sa, ArrayList sb, Object v1,
2656     Object v2)
2657 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2658     sa..contents = sb..contents & sa..csize = sb..csize"
2659     ensures "True" */
2660 {
2661     /*: assume "True" */
2662     int r1a = sa.lastIndexOf(v1);
2663     int r2a = sa.indexOf(v2);
2664
2665     int r2b = sb.indexOf(v2);
2666     int r1b = sb.lastIndexOf(v1);
2667
2668     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2669         sb..csize" */
2670 }
2671
2672 static void lastIndexOf_indexOf_pre_c_87(ArrayList sa, ArrayList sb, Object v1,
2673     Object v2)
2674 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2675     sa..contents = sb..contents & sa..csize = sb..csize"
2676     ensures "True" */
2677 {
2678     /*: assume "~(True)" */
2679     int r1a = sa.lastIndexOf(v1);
2680     int r2a = sa.indexOf(v2);
2681
2682     int r2b = sb.indexOf(v2);
2683     int r1b = sb.lastIndexOf(v1);
2684
2685     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2686         sb..csize)" */
2687 }
2688
2689 static void lastIndexOf_indexOf_between_s_88(ArrayList sa, ArrayList sb, Object v1,
2690     Object v2)
2691 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2692     sa..contents = sb..contents & sa..csize = sb..csize"
2693     ensures "True" */
2694 {
2695     int r1a = sa.lastIndexOf(v1);
2696     /*: assume "True" */
2697     int r2a = sa.indexOf(v2);
2698
2699     int r2b = sb.indexOf(v2);
2700     int r1b = sb.lastIndexOf(v1);
2701
2702     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2703         sb..csize" */
2704 }

```

```

2698 static void lastIndexOf_indexOf_between_c_88(ArrayList sa, ArrayList sb, Object v1,
2699 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 =
2712 sb..csize)" */
2713 }
2714
2715 static void lastIndexOf_indexOf_post_s_89(ArrayList sa, ArrayList sb, Object v1,
2716 Object v2)
2717 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2718 sa..contents = sb..contents & sa..csize = sb..csize"
2719 ensures "True" */
2720 {
2721     int r1a = sa.lastIndexOf(v1);
2722     int r2a = sa.indexOf(v2);
2723     /*: assume "True" */
2724
2725     int r2b = sb.indexOf(v2);
2726     int r1b = sb.lastIndexOf(v1);
2727
2728     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2729 sb..csize" */
2730 }
2731
2732 static void lastIndexOf_indexOf_post_c_89(ArrayList sa, ArrayList sb, Object v1,
2733 Object v2)
2734 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2735 sa..contents = sb..contents & sa..csize = sb..csize"
2736 ensures "True" */
2737 {
2738     int r1a = sa.lastIndexOf(v1);
2739     int r2a = sa.indexOf(v2);
2740     /*: assume "~(True)" */
2741
2742     int r2b = sb.indexOf(v2);
2743     int r1b = sb.lastIndexOf(v1);
2744
2745     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2746 sb..csize)" */
2747 }
2748
2749 static void lastIndexOf_lastIndexOf_pre_s_90(ArrayList sa, ArrayList sb, Object v1,
2750 Object v2)
2751 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2752 sa..contents = sb..contents & sa..csize = sb..csize"
2753 ensures "True" */
2754 {
2755     /*: assume "True" */
2756     int r1a = sa.lastIndexOf(v1);
2757     int r2a = sa.lastIndexOf(v2);
2758
2759     int r2b = sb.lastIndexOf(v2);
2760     int r1b = sb.lastIndexOf(v1);

```

```

2756     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2757         sb..csize" */
2758 }
2759 static void lastIndexOf_lastIndexOf_pre_c_90(ArrayList sa, ArrayList sb, Object v1,
2760     Object v2)
2761 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2762     sa..contents = sb..contents & sa..csize = sb..csize"
2763     ensures "True" */
2764 {
2765     /*: assume "~(True)" */
2766     int r1a = sa.lastIndexOf(v1);
2767     int r2a = sa.lastIndexOf(v2);
2768
2769     int r2b = sb.lastIndexOf(v2);
2770     int r1b = sb.lastIndexOf(v1);
2771
2772     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2773         sb..csize)" */
2774 }
2775 static void lastIndexOf_lastIndexOf_between_s_91(ArrayList sa, ArrayList sb, Object
2776     v1, Object v2)
2777 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2778     sa..contents = sb..contents & sa..csize = sb..csize"
2779     ensures "True" */
2780 {
2781     int r1a = sa.lastIndexOf(v1);
2782     /*: assume "True" */
2783     int r2a = sa.lastIndexOf(v2);
2784
2785     int r2b = sb.lastIndexOf(v2);
2786     int r1b = sb.lastIndexOf(v1);
2787
2788     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2789         sb..csize" */
2790 }
2791 static void lastIndexOf_lastIndexOf_between_c_91(ArrayList sa, ArrayList sb, Object
2792     v1, Object v2)
2793 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2794     sa..contents = sb..contents & sa..csize = sb..csize"
2795     ensures "True" */
2796 {
2797     int r1a = sa.lastIndexOf(v1);
2798     /*: assume "~(True)" */
2799     int r2a = sa.lastIndexOf(v2);
2800
2801     int r2b = sb.lastIndexOf(v2);
2802     int r1b = sb.lastIndexOf(v1);
2803
2804     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2805         sb..csize)" */
2806 }
2807 static void lastIndexOf_lastIndexOf_post_s_92(ArrayList sa, ArrayList sb, Object
2808     v1, Object v2)
2809 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2810     sa..contents = sb..contents & sa..csize = sb..csize"
2811     ensures "True" */
2812 {
2813     int r1a = sa.lastIndexOf(v1);
2814     int r2a = sa.lastIndexOf(v2);
2815     /*: assume "True" */

```

```

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
2870     v1, int i2)
2871 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2872     sa..contents = sb..contents & sa..csize = sb..csize"
2873     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2874     ensures "True" */
2875 {
2876     int r1a = sa.lastIndexOf(v1);
2877     /*: assume "r1a < 0 | (0 <= r1a & r1a < i2)" */
2878     /*: assume "0 <= i2 & i2 < sa..csize" */
2879     Object r2a = sa.remove_at(i2);
2880
2881     Object r2b = sb.remove_at(i2);
2882     int r1b = sb.lastIndexOf(v1);
2883
2884     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2885     sb..csize" */
2886 }
2887
2888 static void lastIndexOf_remove_at_between_c_94(ArrayList sa, ArrayList sb, Object
2889     v1, int i2)
2890 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2891     sa..contents = sb..contents & sa..csize = sb..csize"
2892     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2893     ensures "True" */
2894 {
2895     int r1a = sa.lastIndexOf(v1);
2896     /*: assume "~(r1a < 0 | (0 <= r1a & r1a < i2))" */
2897     /*: assume "0 <= i2 & i2 < sa..csize" */
2898     Object r2a = sa.remove_at(i2);
2899
2900     /*: assume "0 <= i2 & i2 < sb..csize" */
2901     Object r2b = sb.remove_at(i2);
2902     int r1b = sb.lastIndexOf(v1);
2903
2904     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2905     sb..csize)" */
2906 }
2907
2908 static void lastIndexOf_remove_at_post_s_95(ArrayList sa, ArrayList sb, Object v1,
2909     int i2)
2910 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2911     sa..contents = sb..contents & sa..csize = sb..csize"
2912     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2913     ensures "True" */
2914 {
2915     int r1a = sa.lastIndexOf(v1);
2916     /*: assume "0 <= i2 & i2 < sa..csize" */
2917     Object r2a = sa.remove_at(i2);
2918     /*: assume "r1a < 0 | (0 <= r1a & r1a < i2)" */
2919
2920     Object r2b = sb.remove_at(i2);
2921     int r1b = sb.lastIndexOf(v1);
2922
2923     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
2924     sb..csize" */
2925 }
2926
2927 static void lastIndexOf_remove_at_post_c_95(ArrayList sa, ArrayList sb, Object v1,
2928     int i2)
2929 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2930     sa..contents = sb..contents & sa..csize = sb..csize"
2931     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 =
2937         sb..csize)" */
2938 }
2939
2940 static void lastIndexOf_remove_at_pre_s_96(ArrayList sa, ArrayList sb, Object v1,
2941     int i2)
2942 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2943     sa..contents = sb..contents & sa..csize = sb..csize"
2944 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2945 ensures "True" */
2946 {
2947     /*: assume "~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) | ((EX i.
2948         (i, v1) : sa..contents & 0 <= i & i < i2) & ~(EX i. (i, v1) : sa..contents &
2949         i2 <= i & i < sa..csize))" */
2950     int r1a = sa.lastIndexOf(v1);
2951     /*: assume "0 <= i2 & i2 < sa..csize" */
2952     sa.remove_at(i2);
2953
2954     sb.remove_at(i2);
2955     int r1b = sb.lastIndexOf(v1);
2956
2957     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
2958 }
2959
2960 static void lastIndexOf_remove_at_pre_c_96(ArrayList sa, ArrayList sb, Object v1,
2961     int i2)
2962 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2963     sa..contents = sb..contents & sa..csize = sb..csize"
2964 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2965 ensures "True" */
2966 {
2967     /*: assume "~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) | ((EX
2968         i. (i, v1) : sa..contents & 0 <= i & i < i2) & ~(EX i. (i, v1) :
2969         sa..contents & i2 <= i & i < sa..csize))" */
2970     int r1a = sa.lastIndexOf(v1);
2971     /*: assume "0 <= i2 & i2 < sa..csize" */
2972     sa.remove_at(i2);
2973
2974     /*: assume "0 <= i2 & i2 < sb..csize" */
2975     sb.remove_at(i2);
2976     int r1b = sb.lastIndexOf(v1);
2977
2978     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
2979         */
2980 }
2981
2982 static void lastIndexOf_remove_at_between_s_97(ArrayList sa, ArrayList sb, Object
2983     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 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2987 ensures "True" */
2988 {
2989     int r1a = sa.lastIndexOf(v1);

```



```

2981     /*: assume "r1a < 0 | (0 <= r1a & r1a < i2)" */
2982     /*: assume "0 <= i2 & i2 < sa..csize" */
2983     sa.remove_at(i2);
2984
2985     sb.remove_at(i2);
2986     int r1b = sb.lastIndexOf(v1);
2987
2988     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
2989 }
2990
2991 static void lastIndexOf_remove_at_between_c_97(ArrayList sa, ArrayList sb, Object
2992     v1, int i2)
2993 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
2994     sa..contents = sb..contents & sa..csize = sb..csize"
2995     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
2996     ensures "True" */
2997 {
2998     int r1a = sa.lastIndexOf(v1);
2999     /*: assume "~(r1a < 0 | (0 <= r1a & r1a < i2))" */
3000     /*: assume "0 <= i2 & i2 < sa..csize" */
3001     sa.remove_at(i2);
3002
3003     /*: assume "0 <= i2 & i2 < sb..csize" */
3004     sb.remove_at(i2);
3005     int r1b = sb.lastIndexOf(v1);
3006
3007     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
3008     */
3009 }
3010
3011 static void lastIndexOf_remove_at_post_s_98(ArrayList sa, ArrayList sb, Object v1,
3012     int i2)
3013 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3014     sa..contents = sb..contents & sa..csize = sb..csize"
3015     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3016     ensures "True" */
3017 {
3018     int r1a = sa.lastIndexOf(v1);
3019     /*: assume "0 <= i2 & i2 < sa..csize" */
3020     sa.remove_at(i2);
3021     /*: assume "r1a < 0 | (0 <= r1a & r1a < i2)" */
3022
3023     sb.remove_at(i2);
3024     int r1b = sb.lastIndexOf(v1);
3025
3026     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
3027 }
3028
3029 static void lastIndexOf_remove_at_post_c_98(ArrayList sa, ArrayList sb, Object v1,
3030     int i2)
3031 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3032     sa..contents = sb..contents & sa..csize = sb..csize"
3033     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3034     ensures "True" */
3035 {
3036     int r1a = sa.lastIndexOf(v1);
3037     /*: assume "0 <= i2 & i2 < sa..csize" */
3038     sa.remove_at(i2);
3039     /*: assume "~(r1a < 0 | (0 <= r1a & r1a < i2))" */
3040
3041     /*: assume "0 <= i2 & i2 < sb..csize" */
3042     sb.remove_at(i2);
3043     int r1b = sb.lastIndexOf(v1);

```

```

3041     /*: assert "~(ria = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
3042 }
3043
3044 static void lastIndexOf_set_pre_s_99(ArrayList sa, ArrayList sb, Object v1, int i2,
    Object v2)
3045 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3046     sa..contents = sb..contents & sa..csize = sb..csize"
3047     modifies "sa..contents", "sb..contents"
3048     ensures "True" */
3049 {
3050     /*: 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))" */
3051     int r1a = sa.lastIndexOf(v1);
3052     /*: assume "0 <= i2 & i2 < sa..csize" */
3053     Object r2a = sa.set(i2, v2);
3054
3055     Object r2b = sb.set(i2, v2);
3056     int r1b = sb.lastIndexOf(v1);
3057
3058     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
3059 }
3060
3061 static void lastIndexOf_set_pre_c_99(ArrayList sa, ArrayList sb, Object v1, int i2,
    Object v2)
3062 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3063     sa..contents = sb..contents & sa..csize = sb..csize"
3064     modifies "sa..contents", "sb..contents"
3065     ensures "True" */
3066 {
3067     /*: 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))" */
3068     int r1a = sa.lastIndexOf(v1);
3069     /*: assume "0 <= i2 & i2 < sa..csize" */
3070     Object r2a = sa.set(i2, v2);
3071
3072     /*: assume "0 <= i2 & i2 < sb..csize" */
3073     Object r2b = sb.set(i2, v2);
3074     int r1b = sb.lastIndexOf(v1);
3075
3076     /*: assert "~(ria = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
3077 }
3078
3079 static void lastIndexOf_set_between_s_100(ArrayList sa, ArrayList sb, Object v1,
    int i2, Object v2)
3080 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3081     sa..contents = sb..contents & sa..csize = sb..csize"
3082     modifies "sa..contents", "sb..contents"
3083     ensures "True" */
3084 {
3085     int r1a = sa.lastIndexOf(v1);
3086     /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2) | (r1a = i2
        & v1 = v2) | r1a > i2" */
3087     /*: assume "0 <= i2 & i2 < sa..csize" */
3088     Object r2a = sa.set(i2, v2);

```

```

3089     Object r2b = sb.set(i2, v2);
3090     int r1b = sb.lastIndexOf(v1);
3091
3092     /*: assert "ria = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3093         sb..csize" */
3094 }
3095
3096 static void lastIndexOf_set_between_c_100(ArrayList sa, ArrayList sb, Object v1,
3097     int i2, Object v2)
3098 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3099     sa..contents = sb..contents & sa..csize = sb..csize"
3100     modifies "sa..contents", "sb..contents"
3101     ensures "True" */
3102 {
3103     int r1a = sa.lastIndexOf(v1);
3104     /*: assume "~((ria < 0 & v1 ~= v2) | (0 <= ria & ria < i2 & v1 ~= v2) | (ria =
3105         i2 & v1 = v2) | ria > i2)" */
3106     /*: assume "0 <= i2 & i2 < sa..csize" */
3107     Object r2a = sa.set(i2, v2);
3108
3109     /*: assume "0 <= i2 & i2 < sb..csize" */
3110     Object r2b = sb.set(i2, v2);
3111     int r1b = sb.lastIndexOf(v1);
3112
3113     /*: assert "~(ria = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3114         sb..csize)" */
3115 }
3116
3117 static void lastIndexOf_set_post_s_101(ArrayList sa, ArrayList sb, Object v1, int
3118     i2, Object v2)
3119 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3120     sa..contents = sb..contents & sa..csize = sb..csize"
3121     modifies "sa..contents", "sb..contents"
3122     ensures "True" */
3123 {
3124     int r1a = sa.lastIndexOf(v1);
3125     /*: assume "0 <= i2 & i2 < sa..csize" */
3126     Object r2a = sa.set(i2, v2);
3127     /*: assume "(ria < 0 & v1 ~= v2) | (0 <= ria & ria < i2 & v1 ~= v2) | (ria = i2
3128         & v1 = v2) | ria > i2" */
3129
3130     Object r2b = sb.set(i2, v2);
3131     int r1b = sb.lastIndexOf(v1);
3132
3133     /*: assert "ria = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3134         sb..csize" */
3135 }
3136
3137 static void lastIndexOf_set_post_c_101(ArrayList sa, ArrayList sb, Object v1, int
3138     i2, Object v2)
3139 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3140     sa..contents = sb..contents & sa..csize = sb..csize"
3141     modifies "sa..contents", "sb..contents"
3142     ensures "True" */
3143 {
3144     int r1a = sa.lastIndexOf(v1);
3145     /*: assume "0 <= i2 & i2 < sa..csize" */
3146     Object r2a = sa.set(i2, v2);
3147     /*: assume "~((ria < 0 & v1 ~= v2) | (0 <= ria & ria < i2 & v1 ~= v2) | (ria =
3148         i2 & v1 = v2) | ria > i2)" */
3149
3150     /*: assume "0 <= i2 & i2 < sb..csize" */
3151     Object r2b = sb.set(i2, v2);
3152     int r1b = sb.lastIndexOf(v1);

```

```

3145     /*: assert "~(ria = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3146         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 "ria = 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 "~(ria = 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 "(ria < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2) | (ria = i2
        & v1 = v2) | r1a > i2" */
3192     /*: assume "0 <= i2 & i2 < sa..csize" */
3193     sa.set(i2, v2);

```

```

3194     sb.set(i2, v2);
3195     int r1b = sb.lastIndexOf(v1);
3196
3197     /*: assert "ria = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
3198 }
3199
3200
3201 static void lastIndexOf_set_between_c_103(ArrayList sa, ArrayList sb, Object v1,
3202     int i2, Object v2)
3203 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3204     sa..contents = sb..contents & sa..csize = sb..csize"
3205     modifies "sa..contents", "sb..contents"
3206     ensures "True" */
3207 {
3208     int r1a = sa.lastIndexOf(v1);
3209     /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2) | (r1a =
3210     i2 & v1 = v2) | r1a > i2)" */
3211     /*: assume "0 <= i2 & i2 < sa..csize" */
3212     sa.set(i2, v2);
3213
3214     /*: assume "0 <= i2 & i2 < sb..csize" */
3215     sb.set(i2, v2);
3216     int r1b = sb.lastIndexOf(v1);
3217
3218     /*: assert "~(ria = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
3219     */
3220 }
3221
3222 static void lastIndexOf_set_post_s_104(ArrayList sa, ArrayList sb, Object v1, int
3223     i2, Object v2)
3224 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3225     sa..contents = sb..contents & sa..csize = sb..csize"
3226     modifies "sa..contents", "sb..contents"
3227     ensures "True" */
3228 {
3229     int r1a = sa.lastIndexOf(v1);
3230     /*: assume "0 <= i2 & i2 < sa..csize" */
3231     sa.set(i2, v2);
3232     /*: assume "(r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2) | (r1a = i2
3233     & v1 = v2) | r1a > i2" */
3234
3235     sb.set(i2, v2);
3236     int r1b = sb.lastIndexOf(v1);
3237
3238     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
3239 }
3240
3241 static void lastIndexOf_set_post_c_104(ArrayList sa, ArrayList sb, Object v1, int
3242     i2, Object v2)
3243 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3244     sa..contents = sb..contents & sa..csize = sb..csize"
3245     modifies "sa..contents", "sb..contents"
3246     ensures "True" */
3247 {
3248     int r1a = sa.lastIndexOf(v1);
3249     /*: assume "0 <= i2 & i2 < sa..csize" */
3250     sa.set(i2, v2);
3251     /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2) | (r1a =
3252     i2 & v1 = v2) | r1a > i2)" */
3253
3254     /*: assume "0 <= i2 & i2 < sb..csize" */
3255     sb.set(i2, v2);
3256     int r1b = sb.lastIndexOf(v1);

```

```

3251     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
3252         */
3253 }
3254 static void lastIndexOf_size_pre_s_105(ArrayList sa, ArrayList sb, Object v1)
3255 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3256     sa..contents = sb..contents & sa..csize = sb..csize"
3257     ensures "True" */
3258 {
3259     /*: assume "True" */
3260     int r1a = sa.lastIndexOf(v1);
3261     int r2a = sa.size();
3262
3263     int r2b = sb.size();
3264     int r1b = sb.lastIndexOf(v1);
3265
3266     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3267         sb..csize" */
3268 }
3269 static void lastIndexOf_size_pre_c_105(ArrayList sa, ArrayList sb, Object v1)
3270 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3271     sa..contents = sb..contents & sa..csize = sb..csize"
3272     ensures "True" */
3273 {
3274     /*: assume "~(True)" */
3275     int r1a = sa.lastIndexOf(v1);
3276     int r2a = sa.size();
3277
3278     int r2b = sb.size();
3279     int r1b = sb.lastIndexOf(v1);
3280
3281     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3282         sb..csize)" */
3283 }
3284 static void lastIndexOf_size_between_s_106(ArrayList sa, ArrayList sb, Object v1)
3285 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3286     sa..contents = sb..contents & sa..csize = sb..csize"
3287     ensures "True" */
3288 {
3289     int r1a = sa.lastIndexOf(v1);
3290     /*: assume "True" */
3291     int r2a = sa.size();
3292
3293     int r2b = sb.size();
3294     int r1b = sb.lastIndexOf(v1);
3295
3296     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3297         sb..csize" */
3298 }
3299 static void lastIndexOf_size_between_c_106(ArrayList sa, ArrayList sb, Object v1)
3300 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3301     sa..contents = sb..contents & sa..csize = sb..csize"
3302     ensures "True" */
3303 {
3304     int r1a = sa.lastIndexOf(v1);
3305     /*: assume "~(True)" */
3306     int r2a = sa.size();
3307
3308     int r2b = sb.size();
3309     int r1b = sb.lastIndexOf(v1);
3310 }

```

```

3311     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3312         sb..csize)" */
3313 }
3314 static void lastIndexOf_size_post_s_107(ArrayList sa, ArrayList sb, Object v1)
3315 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3316     sa..contents = sb..contents & sa..csize = sb..csize"
3317     ensures "True" */
3318 {
3319     int r1a = sa.lastIndexOf(v1);
3320     int r2a = sa.size();
3321     /*: assume "True" */
3322
3323     int r2b = sb.size();
3324     int r1b = sb.lastIndexOf(v1);
3325
3326     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3327         sb..csize" */
3328 }
3329 static void lastIndexOf_size_post_c_107(ArrayList sa, ArrayList sb, Object v1)
3330 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3331     sa..contents = sb..contents & sa..csize = sb..csize"
3332     ensures "True" */
3333 {
3334     int r1a = sa.lastIndexOf(v1);
3335     int r2a = sa.size();
3336     /*: assume "~(True)" */
3337
3338     int r2b = sb.size();
3339     int r1b = sb.lastIndexOf(v1);
3340
3341     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3342         sb..csize)" */
3343 }
3344 static void remove_at_add_at_pre_s_108(ArrayList sa, ArrayList sb, int i1, int i2,
3345     Object v2)
3346 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3347     sa..contents = sb..contents & sa..csize = sb..csize"
3348     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
3349     "sb..msize"
3350     ensures "True" */
3351 {
3352     /*: assume "(i1 < i2 & (i2, v2) : sa..contents & 0 <= i2 & i2 < sa..csize) |
3353         (i1 = i2 & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize) | (i1 > i2 &
3354         (ALL v. ((i1 - 1, v) : sa..contents) = ((i1, v) : sa..contents)) & 0 <= i1 -
3355         1 & i1 - 1 < sa..csize & 0 <= i1 & i1 < sa..csize)" */
3356     /*: assume "0 <= i1 & i1 < sa..csize" */
3357     Object r1a = sa.remove_at(i1);
3358     /*: assume "0 <= i2 & i2 <= sa..csize" */
3359     sa.add_at(i2, v2);
3360
3361     sb.add_at(i2, v2);
3362     Object r1b = sb.remove_at(i1);
3363
3364     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
3365 }
3366 static void remove_at_add_at_pre_c_108(ArrayList sa, ArrayList sb, int i1, int i2,
3367     Object v2)
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", "sa..msize",
3371     "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 &
        sa..contents = sb..contents & sa..csize = sb..csize"
3384     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
        "sb..msize"
3385     ensures "True" */
3386 {
3387     /*: assume "0 <= i1 & i1 < sa..csize" */
3388     Object r1a = sa.remove_at(i1);
3389     /*: 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)" */
3390     /*: assume "0 <= i2 & i2 <= sa..csize" */
3391     sa.add_at(i2, v2);
3392
3393     sb.add_at(i2, v2);
3394     Object r1b = sb.remove_at(i1);
3395
3396     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
3397 }
3398
3399 static void remove_at_add_at_between_c_109(ArrayList sa, ArrayList sb, int i1, int
        i2, Object v2)
3400 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
3401     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
        "sb..msize"
3402     ensures "True" */
3403 {
3404     /*: assume "0 <= i1 & i1 < sa..csize" */
3405     Object r1a = sa.remove_at(i1);
3406     /*: 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))" */
3407     /*: assume "0 <= i2 & i2 <= sa..csize" */
3408     sa.add_at(i2, v2);
3409
3410     sb.add_at(i2, v2);
3411     Object r1b = sb.remove_at(i1);
3412
3413     /*: assume "0 <= i2 & i2 <= sb..csize" */
3414     sb.add_at(i2, v2);
3415     /*: assume "0 <= i1 & i1 < sb..csize" */
3416     Object r1b = sb.remove_at(i1);
3417
3418     /*: 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

```

```

3470     Object r2b = sb.get(i2);
3471     Object r1b = sb.remove_at(i1);
3472
3473     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
          sb..csize" */
3474 }
3475
3476 static void remove_at_get_pre_c_111(ArrayList sa, ArrayList sb, int i1, int i2)
3477 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
          sa..contents = sb..contents & sa..csize = sb..csize"
3478 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3479 ensures "True" */
3480 {
3481     /*: 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)" */
3482     /*: assume "0 <= i1 & i1 < sa..csize" */
3483     Object r1a = sa.remove_at(i1);
3484     /*: assume "0 <= i2 & i2 < sa..csize" */
3485     Object r2a = sa.get(i2);
3486
3487     /*: assume "0 <= i2 & i2 < sb..csize" */
3488     Object r2b = sb.get(i2);
3489     /*: assume "0 <= i1 & i1 < sb..csize" */
3490     Object r1b = sb.remove_at(i1);
3491
3492     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
          sb..csize)" */
3493 }
3494
3495 static void remove_at_get_between_s_112(ArrayList sa, ArrayList sb, int i1, int i2)
3496 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
          sa..contents = sb..contents & sa..csize = sb..csize"
3497 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3498 ensures "True" */
3499 {
3500     /*: assume "0 <= i1 & i1 < sa..csize" */
3501     Object r1a = sa.remove_at(i1);
3502     /*: 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" */
3503     /*: assume "0 <= i2 & i2 < sa..csize" */
3504     Object r2a = sa.get(i2);
3505
3506     Object r2b = sb.get(i2);
3507     Object r1b = sb.remove_at(i1);
3508
3509     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
          sb..csize" */
3510 }
3511
3512 static void remove_at_get_between_c_112(ArrayList sa, ArrayList sb, int i1, int i2)
3513 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
          sa..contents = sb..contents & sa..csize = sb..csize"
3514 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3515 ensures "True" */
3516 {
3517     /*: assume "0 <= i1 & i1 < sa..csize" */
3518     Object r1a = sa.remove_at(i1);
3519     /*: 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)" */
3520     /*: assume "0 <= i2 & i2 < sa..csize" */
3521 }
3522
3523

```

```

3524     Object r2a = sa.get(i2);
3525
3526     /*: assume "0 <= i2 & i2 < sb..csize" */
3527     Object r2b = sb.get(i2);
3528     /*: assume "0 <= i1 & i1 < sb..csize" */
3529     Object r1b = sb.remove_at(i1);
3530
3531     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3532         sb..csize)" */
3533 }
3534
3535 static void remove_at_get_post_s_113(ArrayList sa, ArrayList sb, int i1, int i2)
3536 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3537     sa..contents = sb..contents & sa..csize = sb..csize"
3538 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3539 ensures "True" */
3540 {
3541     /*: assume "0 <= i1 & i1 < sa..csize" */
3542     Object r1a = sa.remove_at(i1);
3543     /*: assume "0 <= i2 & i2 < sa..csize" */
3544     Object r2a = sa.get(i2);
3545     /*: assume "(i1 < i2 & (i2 - 1, r2a) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
3546         sa..csize) | (i1 = i2 & r1a = r2a) | i1 > i2" */
3547
3548     Object r2b = sb.get(i2);
3549     Object r1b = sb.remove_at(i1);
3550
3551     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3552         sb..csize" */
3553 }
3554
3555 static void remove_at_get_post_c_113(ArrayList sa, ArrayList sb, int i1, int i2)
3556 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3557     sa..contents = sb..contents & sa..csize = sb..csize"
3558 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3559 ensures "True" */
3560 {
3561     /*: assume "0 <= i1 & i1 < sa..csize" */
3562     Object r1a = sa.remove_at(i1);
3563     /*: assume "0 <= i2 & i2 < sa..csize" */
3564     Object r2a = sa.get(i2);
3565     /*: assume "~((i1 < i2 & (i2 - 1, r2a) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
3566         sa..csize) | (i1 = i2 & r1a = r2a) | i1 > i2)" */
3567
3568     /*: assume "0 <= i2 & i2 < sb..csize" */
3569     Object r2b = sb.get(i2);
3570     /*: assume "0 <= i1 & i1 < sb..csize" */
3571     Object r1b = sb.remove_at(i1);
3572
3573     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3574         sb..csize)" */
3575 }
3576
3577 static void remove_at_lastIndexOf_pre_s_117(ArrayList sa, ArrayList sb, int i1,
3578     Object v2)
3579 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3580     sa..contents = sb..contents & sa..csize = sb..csize"
3581 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3582 ensures "True" */
3583 {
3584     /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | ((EX i.
3585         (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) : sa..contents &
3586         i1 <= i & i < sa..csize))" */
3587     /*: assume "0 <= i1 & i1 < sa..csize" */
3588     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 &
        sa..contents = sb..contents & sa..csize = sb..csize"
3591     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3592     ensures "True" */
3593 {
3594     /*: 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)))" */
3595     /*: assume "0 <= i1 & i1 < sa..csize" */
3596     Object r1a = sa.remove_at(i1);
3597     int r2a = sa.lastIndexOf(v2);
3598
3599     int r2b = sb.lastIndexOf(v2);
3600     /*: assume "0 <= i1 & i1 < sb..csize" */
3601     Object r1b = sb.remove_at(i1);
3602
3603     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
3604 }
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 &
        sa..contents = sb..contents & sa..csize = sb..csize"
3609     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3610     ensures "True" */
3611 {
3612     /*: assume "0 <= i1 & i1 < sa..csize" */
3613     Object r1a = sa.remove_at(i1);
3614     /*: 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))" */
3615     int r2a = sa.lastIndexOf(v2);
3616
3617     int r2b = sb.lastIndexOf(v2);
3618     /*: assume "0 <= i1 & i1 < sb..csize" */
3619     Object r1b = sb.remove_at(i1);
3620
3621     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
3622 }
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 &
        sa..contents = sb..contents & sa..csize = sb..csize"
3627     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3628     ensures "True" */
3629 {
3630     /*: assume "0 <= i1 & i1 < sa..csize" */
3631     Object r1a = sa.remove_at(i1);
3632     int r2a = sa.lastIndexOf(v2);
3633     /*: assume "~((r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1 & r1a ~= v2))" */
3634 }
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 =
3641         sb..csize)" */
3642 }
3643
3644 static void remove_at_remove_at_pre_s_120(ArrayList sa, ArrayList sb, int i1, int
3645     i2)
3646 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3647     sa..contents = sb..contents & sa..csize = sb..csize"
3648     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3649     ensures "True" */
3650 {
3651     /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
3652         sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
3653         sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :
3654         sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
3655         sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
3656         sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
3657         <= i1 + 1 & i1 + 1 < sa..csize)" */
3658     /*: assume "0 <= i1 & i1 < sa..csize" */
3659     Object r1a = sa.remove_at(i1);
3660     /*: assume "0 <= i2 & i2 < sa..csize" */
3661     Object r2a = sa.remove_at(i2);
3662
3663     Object r2b = sb.remove_at(i2);
3664     Object r1b = sb.remove_at(i1);
3665
3666     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3667         sb..csize" */
3668 }
3669
3670 static void remove_at_remove_at_pre_c_120(ArrayList sa, ArrayList sb, int i1, int
3671     i2)
3672 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3673     sa..contents = sb..contents & sa..csize = sb..csize"
3674     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3675     ensures "True" */
3676 {
3677     /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
3678         sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
3679         sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :
3680         sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
3681         sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
3682         sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
3683         <= i1 + 1 & i1 + 1 < sa..csize))" */
3684     /*: assume "0 <= i1 & i1 < sa..csize" */
3685     Object r1a = sa.remove_at(i1);
3686     /*: assume "0 <= i2 & i2 < sa..csize" */
3687     Object r2a = sa.remove_at(i2);
3688
3689     /*: assume "0 <= i2 & i2 < sb..csize" */
3690     Object r2b = sb.remove_at(i2);
3691     /*: assume "0 <= i1 & i1 < sb..csize" */
3692     Object r1b = sb.remove_at(i1);
3693
3694     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3695         sb..csize)" */
3696 }
3697
3698 static void remove_at_remove_at_between_s_121(ArrayList sa, ArrayList sb, int i1,
3699     int i2)
3700 /*: 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) :
3690         sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
3691         sa..csize) | (i1 = i2 & (i2, r1a) : sa..contents & 0 <= i2 & i2 < sa..csize)
3692         | (sa..csize > i1 & i1 > i2 & (i1, r1a) : sa..contents & 0 <= i1 & i1 <
3693         sa..csize)" */
3694     /*: assume "0 <= i2 & i2 < sa..csize" */
3695     Object r2a = sa.remove_at(i2);
3696
3697     Object r2b = sb.remove_at(i2);
3698     Object r1b = sb.remove_at(i1);
3699
3700     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3701         sb..csize" */
3702 }
3703
3704 static void remove_at_remove_at_between_c_121(ArrayList sa, ArrayList sb, int i1,
3705     int i2)
3706 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3707     sa..contents = sb..contents & sa..csize = sb..csize"
3708 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3709 ensures "True" */
3710 {
3711     /*: assume "0 <= i1 & i1 < sa..csize" */
3712     Object r1a = sa.remove_at(i1);
3713     /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
3714         sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
3715         sa..csize) | (i1 = i2 & (i2, r1a) : sa..contents & 0 <= i2 & i2 < sa..csize)
3716         | (sa..csize > i1 & i1 > i2 & (i1, r1a) : sa..contents & 0 <= i1 & i1 <
3717         sa..csize))" */
3718     /*: assume "0 <= i2 & i2 < sa..csize" */
3719     Object r2a = sa.remove_at(i2);
3720
3721     /*: assume "0 <= i2 & i2 < sb..csize" */
3722     Object r2b = sb.remove_at(i2);
3723     /*: assume "0 <= i1 & i1 < sb..csize" */
3724     Object r1b = sb.remove_at(i1);
3725
3726     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3727         sb..csize)" */
3728 }
3729
3730 static void remove_at_remove_at_post_s_122(ArrayList sa, ArrayList sb, int i1, int
3731     i2)
3732 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3733     sa..contents = sb..contents & sa..csize = sb..csize"
3734 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3735 ensures "True" */
3736 {
3737     /*: assume "0 <= i1 & i1 < sa..csize" */
3738     Object r1a = sa.remove_at(i1);
3739     /*: assume "0 <= i2 & i2 < sa..csize" */
3740     Object r2a = sa.remove_at(i2);
3741     /*: assume "(i1 < i2 & (i2 - 1, r2a) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
3742         sa..csize) | (i1 = i2 & r1a = r2a) | (sa..csize + 1 > i1 & i1 > i2 & (i1 -
3743         1, r1a) : sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize)" */
3744
3745     Object r2b = sb.remove_at(i2);
3746     Object r1b = sb.remove_at(i1);
3747
3748 }

```

```

3734     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3735         sb..csize" */
3736 }
3737
3738 static void remove_at_remove_at_post_c_122(ArrayList sa, ArrayList sb, int i1, int
3739     i2)
3740 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3741     sa..contents = sb..contents & sa..csize = sb..csize"
3742     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3743     ensures "True" */
3744 {
3745     /*: assume "0 <= i1 & i1 < sa..csize" */
3746     Object r1a = sa.remove_at(i1);
3747     /*: assume "0 <= i2 & i2 < sa..csize" */
3748     Object r2a = sa.remove_at(i2);
3749     /*: assume "~((i1 < i2 & (i2 - 1, r2a) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
3750         sa..csize) | (i1 = i2 & r1a = r2a) | (sa..csize + 1 > i1 & i1 > i2 & (i1 -
3751         1, r1a) : sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize))" */
3752
3753     /*: assume "0 <= i2 & i2 < sb..csize" */
3754     Object r2b = sb.remove_at(i2);
3755     /*: assume "0 <= i1 & i1 < sb..csize" */
3756     Object r1b = sb.remove_at(i1);
3757
3758     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3759         sb..csize)" */
3760 }
3761
3762 static void remove_at_remove_at_pre_s_123(ArrayList sa, ArrayList sb, int i1, int
3763     i2)
3764 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3765     sa..contents = sb..contents & sa..csize = sb..csize"
3766     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3767     ensures "True" */
3768 {
3769     /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
3770         sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
3771         sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :
3772         sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
3773         sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
3774         sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
3775         <= i1 + 1 & i1 + 1 < sa..csize)" */
3776     /*: assume "0 <= i1 & i1 < sa..csize" */
3777     Object r1a = sa.remove_at(i1);
3778     /*: assume "0 <= i2 & i2 < sa..csize" */
3779     sa.remove_at(i2);
3780
3781     sb.remove_at(i2);
3782     Object r1b = sb.remove_at(i1);
3783
3784     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
3785 }
3786
3787 static void remove_at_remove_at_pre_c_123(ArrayList sa, ArrayList sb, int i1, int
3788     i2)
3789 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3790     sa..contents = sb..contents & sa..csize = sb..csize"
3791     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3792     ensures "True" */
3793 {
3794     /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
3795         sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
3796         sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :
3797         sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
3798         sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :

```

```

3782     sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
3783     <= i1 + 1 & i1 + 1 < sa..csize))" */
3784 /*: assume "0 <= i1 & i1 < sa..csize" */
3785 Object r1a = sa.remove_at(i1);
3786 /*: assume "0 <= i2 & i2 < sa..csize" */
3787 sa.remove_at(i2);
3788
3789 /*: assume "0 <= i2 & i2 < sb..csize" */
3790 sb.remove_at(i2);
3791 /*: assume "0 <= i1 & i1 < sb..csize" */
3792 Object r1b = sb.remove_at(i1);
3793
3794 /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
3795     */
3796 }
3797
3798 static void remove_at_remove_at_between_s_124(ArrayList sa, ArrayList sb, int i1,
3799 int i2)
3800 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3801     sa..contents = sb..contents & sa..csize = sb..csize"
3802     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3803     ensures "True" */
3804 {
3805     /*: assume "0 <= i1 & i1 < sa..csize" */
3806     Object r1a = sa.remove_at(i1);
3807     /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
3808     sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
3809     sa..csize) | (i1 = i2 & (i2, r1a) : sa..contents & 0 <= i2 & i2 < sa..csize)
3810     | (sa..csize > i1 & i1 > i2 & (i1, r1a) : sa..contents & 0 <= i1 & i1 <
3811     sa..csize)" */
3812     /*: assume "0 <= i2 & i2 < sa..csize" */
3813     sa.remove_at(i2);
3814
3815     sb.remove_at(i2);
3816     Object r1b = sb.remove_at(i1);
3817
3818     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
3819 }
3820
3821 static void remove_at_remove_at_between_c_124(ArrayList sa, ArrayList sb, int i1,
3822 int i2)
3823 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3824     sa..contents = sb..contents & sa..csize = sb..csize"
3825     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3826     ensures "True" */
3827 {
3828     /*: assume "0 <= i1 & i1 < sa..csize" */
3829     Object r1a = sa.remove_at(i1);
3830     /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
3831     sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
3832     sa..csize) | (i1 = i2 & (i2, r1a) : sa..contents & 0 <= i2 & i2 < sa..csize)
3833     | (sa..csize > i1 & i1 > i2 & (i1, r1a) : sa..contents & 0 <= i1 & i1 <
3834     sa..csize))" */
3835     /*: assume "0 <= i2 & i2 < sa..csize" */
3836     sa.remove_at(i2);
3837
3838     /*: assume "0 <= i2 & i2 < sb..csize" */
3839     sb.remove_at(i2);
3840     /*: assume "0 <= i1 & i1 < sb..csize" */
3841     Object r1b = sb.remove_at(i1);
3842
3843     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
3844     */
3845 }
3846
3847 }
3848
3849 }

```



```

3833 static void remove_at_remove_at_post_s_125(ArrayList sa, ArrayList sb, int i1, int
3834 i2)
3835 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3836 sa..contents = sb..contents & sa..csize = sb..csize"
3837 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3838 ensures "True" */
3839 {
3840 /*: assume "0 <= i1 & i1 < sa..csize" */
3841 Object r1a = sa.remove_at(i1);
3842 /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
3843 /*: ghost specvar sa__csize :: "int" = "sa..csize" */
3844 /*: assume "0 <= i2 & i2 < sa..csize" */
3845 sa.remove_at(i2);
3846 /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
3847 sa__contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
3848 sa__csize) | (i1 = i2 & (i2, r1a) : sa__contents & 0 <= i2 & i2 < sa__csize)
3849 | (sa..csize + 1 > i1 & i1 > i2 & (i1 - 1, r1a) : sa..contents & 0 <= i1 - 1
3850 & i1 - 1 < sa..csize)" */
3851
3852 sb.remove_at(i2);
3853 Object r1b = sb.remove_at(i1);
3854
3855 /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
3856 }
3857
3858 static void remove_at_remove_at_post_c_125(ArrayList sa, ArrayList sb, int i1, int
3859 i2)
3860 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3861 sa..contents = sb..contents & sa..csize = sb..csize"
3862 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3863 ensures "True" */
3864 {
3865 /*: assume "0 <= i1 & i1 < sa..csize" */
3866 Object r1a = sa.remove_at(i1);
3867 /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
3868 /*: ghost specvar sa__csize :: "int" = "sa..csize" */
3869 /*: assume "0 <= i2 & i2 < sa..csize" */
3870 sa.remove_at(i2);
3871 /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
3872 sa__contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
3873 sa__csize) | (i1 = i2 & (i2, r1a) : sa__contents & 0 <= i2 & i2 < sa__csize)
3874 | (sa..csize + 1 > i1 & i1 > i2 & (i1 - 1, r1a) : sa..contents & 0 <= i1 - 1
3875 & i1 - 1 < sa..csize))" */
3876
3877 /*: assume "0 <= i2 & i2 < sb..csize" */
3878 sb.remove_at(i2);
3879 /*: assume "0 <= i1 & i1 < sb..csize" */
3880 Object r1b = sb.remove_at(i1);
3881
3882 /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
3883 */
3884 }
3885
3886 static void remove_at_set_pre_s_126(ArrayList sa, ArrayList sb, int i1, int i2,
3887 Object v2)
3888 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3889 sa..contents = sb..contents & sa..csize = sb..csize"
3890 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3891 ensures "True" */
3892 {
3893 /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
3894 sa__contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) : sa..contents & 0
3895 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & (ALL
3896 v. ((i1, v) : sa..contents) = ((i2 + 1, v) : sa..contents)) & (i1, v2) :

```

```

3882     sa..contents & (i2 + 1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <=
3883     i2 + 1 & i2 + 1 < sa..csize) | (i1 > i2)" */
3884 /*: assume "0 <= i1 & i1 < sa..csize" */
3885 Object r1a = sa.remove_at(i1);
3886 /*: assume "0 <= i2 & i2 < sa..csize" */
3887 Object r2a = sa.set(i2, v2);
3888
3889 Object r2b = sb.set(i2, v2);
3890 Object r1b = sb.remove_at(i1);
3891
3892 /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3893     sb..csize" */
3894 }
3895
3896 static void remove_at_set_pre_c_126(ArrayList sa, ArrayList sb, int i1, int i2,
3897     Object v2)
3898 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3899     sa..contents = sb..contents & sa..csize = sb..csize"
3900 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3901 ensures "True" */
3902 {
3903     /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
3904     sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) : sa..contents & 0
3905     <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & (ALL
3906     v. ((i1, v) : sa..contents) = ((i2 + 1, v) : sa..contents)) & (i1, v2) :
3907     sa..contents & (i2 + 1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <=
3908     i2 + 1 & i2 + 1 < sa..csize) | (i1 > i2))" */
3909     /*: assume "0 <= i1 & i1 < sa..csize" */
3910     Object r1a = sa.remove_at(i1);
3911     /*: assume "0 <= i2 & i2 < sa..csize" */
3912     Object r2a = sa.set(i2, v2);
3913
3914     /*: assume "0 <= i2 & i2 < sb..csize" */
3915     Object r2b = sb.set(i2, v2);
3916     /*: assume "0 <= i1 & i1 < sb..csize" */
3917     Object r1b = sb.remove_at(i1);
3918
3919     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3920     sb..csize)" */
3921 }
3922
3923 static void remove_at_set_between_s_127(ArrayList sa, ArrayList sb, int i1, int i2,
3924     Object v2)
3925 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3926     sa..contents = sb..contents & sa..csize = sb..csize"
3927 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3928 ensures "True" */
3929 {
3930     /*: assume "0 <= i1 & i1 < sa..csize" */
3931     Object r1a = sa.remove_at(i1);
3932     /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
3933     sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa..contents & 0
3934     <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize) | (i1 = i2 & (i2,
3935     r1a) : sa..contents & r1a = v2 & (i2, v2) : sa..contents & 0 <= i2 & i2 <
3936     sa..csize) | i1 > i2" */
3937     /*: assume "0 <= i2 & i2 < sa..csize" */
3938     Object r2a = sa.set(i2, v2);
3939
3940     Object r2b = sb.set(i2, v2);
3941     Object r1b = sb.remove_at(i1);
3942
3943     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3944     sb..csize" */
3945 }
3946
3947 }
3948
3949 }
3950

```

```

3931 static void remove_at_set_between_c_127(ArrayList sa, ArrayList sb, int i1, int i2,
3932     Object v2)
3933 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3934     sa..contents = sb..contents & sa..csize = sb..csize"
3935     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3936     ensures "True" */
3937 {
3938     /*: assume "0 <= i1 & i1 < sa..csize" */
3939     Object r1a = sa.remove_at(i1);
3940     /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
3941     sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa..contents & 0
3942     <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize) | (i1 = i2 & (i2,
3943     r1a) : sa..contents & r1a = v2 & (i2, v2) : sa..contents & 0 <= i2 & i2 <
3944     sa..csize) | i1 > i2)" */
3945     /*: assume "0 <= i2 & i2 < sa..csize" */
3946     Object r2a = sa.set(i2, v2);
3947
3948     /*: assume "0 <= i2 & i2 < sb..csize" */
3949     Object r2b = sb.set(i2, v2);
3950     /*: assume "0 <= i1 & i1 < sb..csize" */
3951     Object r1b = sb.remove_at(i1);
3952
3953     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3954     sb..csize)" */
3955 }
3956
3957 static void remove_at_set_post_s_128(ArrayList sa, ArrayList sb, int i1, int i2,
3958     Object v2)
3959 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3960     sa..contents = sb..contents & sa..csize = sb..csize"
3961     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3962     ensures "True" */
3963 {
3964     /*: assume "0 <= i1 & i1 < sa..csize" */
3965     Object r1a = sa.remove_at(i1);
3966     /*: assume "0 <= i2 & i2 < sa..csize" */
3967     Object r2a = sa.set(i2, v2);
3968     /*: assume "(i1 < i2 & (i2 - 1, r2a) : sa..contents & (i2 - 1, v2) :
3969     sa..contents & r2a = v2 & 0 <= i2 - 1 & i2 - 1 < sa..csize) | (i1 = i2 & r1a
3970     = r2a & r1a = v2 & r2a = v2) | i1 > i2" */
3971
3972     Object r2b = sb.set(i2, v2);
3973     Object r1b = sb.remove_at(i1);
3974
3975     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3976     sb..csize" */
3977 }
3978
3979 static void remove_at_set_post_c_128(ArrayList sa, ArrayList sb, int i1, int i2,
3980     Object v2)
3981 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3982     sa..contents = sb..contents & sa..csize = sb..csize"
3983     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3984     ensures "True" */
3985 {
3986     /*: assume "0 <= i1 & i1 < sa..csize" */
3987     Object r1a = sa.remove_at(i1);
3988     /*: assume "0 <= i2 & i2 < sa..csize" */
3989     Object r2a = sa.set(i2, v2);
3990     /*: assume "~((i1 < i2 & (i2 - 1, r2a) : sa..contents & (i2 - 1, v2) :
3991     sa..contents & r2a = v2 & 0 <= i2 - 1 & i2 - 1 < sa..csize) | (i1 = i2 & r1a
3992     = r2a & r1a = v2 & r2a = v2) | i1 > i2)" */
3993
3994     /*: assume "0 <= i2 & i2 < sb..csize" */
3995     Object r2b = sb.set(i2, v2);

```

```

3983     /*: assume "0 <= i1 & i1 < sb..csize" */
3984     Object r1b = sb.remove_at(i1);
3985
3986     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
3987         sb..csize)" */
3988 }
3989
3990 static void remove_at_set_pre_s_129(ArrayList sa, ArrayList sb, int i1, int i2,
3991     Object v2)
3992 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
3993     sa..contents = sb..contents & sa..csize = sb..csize"
3994 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
3995 ensures "True" */
3996 {
3997     /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
3998     sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) : sa..contents & 0
3999     <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & (ALL
4000     v. ((i1, v) : sa..contents) = ((i2 + 1, v) : sa..contents)) & (i1, v2) :
4001     sa..contents & (i2 + 1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <=
4002     i2 + 1 & i2 + 1 < sa..csize) | (i1 > i2)" */
4003     /*: assume "0 <= i1 & i1 < sa..csize" */
4004     Object r1a = sa.remove_at(i1);
4005     /*: assume "0 <= i2 & i2 < sa..csize" */
4006     sa.set(i2, v2);
4007
4008     sb.set(i2, v2);
4009     Object r1b = sb.remove_at(i1);
4010
4011     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
4012 }
4013
4014 static void remove_at_set_between_s_130(ArrayList sa, ArrayList sb, int i1, int i2,
4015     Object v2)
4016 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4017     sa..contents = sb..contents & sa..csize = sb..csize"
4018 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4019 ensures "True" */
4020 {
4021     /*: assume "0 <= i1 & i1 < sa..csize" */
4022     Object r1a = sa.remove_at(i1);
4023     /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
4024     sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa..contents & 0
4025     <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize) | (i1 = i2 & (i2,
4026     r1a) : sa..contents & r1a = v2 & (i2, v2) : sa..contents & 0 <= i2 & i2 <
4027     sa..csize) | i1 > i2" */
4028     /*: assume "0 <= i2 & i2 < sa..csize" */
4029     sa.set(i2, v2);
4030
4031     sb.set(i2, v2);
4032     Object r1b = sb.remove_at(i1);
4033
4034     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
4035 }
4036
4037 static void remove_at_set_between_c_130(ArrayList sa, ArrayList sb, int i1, int i2,
4038     Object v2)
4039 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4040     sa..contents = sb..contents & sa..csize = sb..csize"
4041 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4042 ensures "True" */
4043 {
4044     /*: assume "0 <= i1 & i1 < sa..csize" */
4045     Object r1a = sa.remove_at(i1);
4046     /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
4047     sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa..contents & 0

```

```

4034     <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize) | (i1 = i2 & (i2,
4035     r1a) : sa..contents & r1a = v2 & (i2, v2) : sa..contents & 0 <= i2 & i2 <
4036     sa..csize) | i1 > i2)" */
4037     /*: assume "0 <= i2 & i2 < sa..csize" */
4038     sa.set(i2, v2);
4039
4040     /*: assume "0 <= i2 & i2 < sb..csize" */
4041     sb.set(i2, v2);
4042     /*: assume "0 <= i1 & i1 < sb..csize" */
4043     Object r1b = sb.remove_at(i1);
4044
4045     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
4046     */
4047 }
4048
4049 static void remove_at_set_post_s_131(ArrayList sa, ArrayList sb, int i1, int i2,
4050     Object v2)
4051 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4052     sa..contents = sb..contents & sa..csize = sb..csize"
4053     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4054     ensures "True" */
4055 {
4056     /*: assume "0 <= i1 & i1 < sa..csize" */
4057     Object r1a = sa.remove_at(i1);
4058     /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
4059     /*: ghost specvar sa__csize :: "int" = "sa..csize" */
4060     /*: assume "0 <= i2 & i2 < sa..csize" */
4061     sa.set(i2, v2);
4062     /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
4063     sa__contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa__contents & 0
4064     <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa__csize) | (i1 = i2 & (i2,
4065     r1a) : sa__contents & r1a = v2 & (i2, v2) : sa__contents & 0 <= i2 & i2 <
4066     sa__csize) | i1 > i2" */
4067
4068     sb.set(i2, v2);
4069     Object r1b = sb.remove_at(i1);
4070
4071     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
4072 }
4073
4074 static void remove_at_set_post_c_131(ArrayList sa, ArrayList sb, int i1, int i2,
4075     Object v2)
4076 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4077     sa..contents = sb..contents & sa..csize = sb..csize"
4078     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4079     ensures "True" */
4080 {
4081     /*: assume "0 <= i1 & i1 < sa..csize" */
4082     Object r1a = sa.remove_at(i1);
4083     /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
4084     /*: ghost specvar sa__csize :: "int" = "sa..csize" */
4085     /*: assume "0 <= i2 & i2 < sa..csize" */
4086     sa.set(i2, v2);
4087     /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
4088     sa__contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa__contents & 0
4089     <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa__csize) | (i1 = i2 & (i2,
4090     r1a) : sa__contents & r1a = v2 & (i2, v2) : sa__contents & 0 <= i2 & i2 <
4091     sa__csize) | i1 > i2)" */
4092
4093     /*: assume "0 <= i2 & i2 < sb..csize" */
4094     sb.set(i2, v2);
4095     /*: assume "0 <= i1 & i1 < sb..csize" */
4096     Object r1b = sb.remove_at(i1);
4097

```

```

4084     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
4085 }
4086
4087 static void remove_at_size_pre_s_132(ArrayList sa, ArrayList sb, int i1)
4088 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4089     sa..contents = sb..contents & sa..csize = sb..csize"
4090     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4091     ensures "True" */
4092 {
4093     /*: assume "False" */
4094     /*: assume "0 <= i1 & i1 < sa..csize" */
4095     Object r1a = sa.remove_at(i1);
4096     int r2a = sa.size();
4097
4098     int r2b = sb.size();
4099     Object r1b = sb.remove_at(i1);
4100
4101     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
4102 }
4103
4104 static void remove_at_size_pre_c_132(ArrayList sa, ArrayList sb, int i1)
4105 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4106     sa..contents = sb..contents & sa..csize = sb..csize"
4107     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4108     ensures "True" */
4109 {
4110     /*: assume "~(False)" */
4111     /*: assume "0 <= i1 & i1 < sa..csize" */
4112     Object r1a = sa.remove_at(i1);
4113     int r2a = sa.size();
4114
4115     int r2b = sb.size();
4116     /*: assume "0 <= i1 & i1 < sb..csize" */
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_size_between_s_133(ArrayList sa, ArrayList sb, int i1)
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 "0 <= i1 & i1 < sa..csize" */
4129     Object r1a = sa.remove_at(i1);
4130     /*: assume "False" */
4131     int r2a = sa.size();
4132
4133     int r2b = sb.size();
4134     Object r1b = sb.remove_at(i1);
4135
4136     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
4137 }
4138
4139 static void remove_at_size_between_c_133(ArrayList sa, ArrayList sb, int i1)
4140 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4141     sa..contents = sb..contents & sa..csize = sb..csize"
4142     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4143     ensures "True" */
4144 {

```

```

4145     /*: assume "0 <= i1 & i1 < sa..csize" */
4146     Object r1a = sa.remove_at(i1);
4147     /*: assume "~(False)" */
4148     int r2a = sa.size();
4149
4150     int r2b = sb.size();
4151     /*: assume "0 <= i1 & i1 < sb..csize" */
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_size_post_s_134(ArrayList sa, ArrayList sb, int i1)
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     int r2a = sa.size();
4166     /*: assume "False" */
4167
4168     int r2b = sb.size();
4169     Object r1b = sb.remove_at(i1);
4170
4171     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
         sb..csize" */
4172 }
4173
4174 static void remove_at_size_post_c_134(ArrayList sa, ArrayList sb, int i1)
4175 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4176     sa..contents = sb..contents & sa..csize = sb..csize"
4177     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4178     ensures "True" */
4179 {
4180     /*: assume "0 <= i1 & i1 < sa..csize" */
4181     Object r1a = sa.remove_at(i1);
4182     int r2a = sa.size();
4183     /*: assume "~(False)" */
4184
4185     int r2b = sb.size();
4186     /*: assume "0 <= i1 & i1 < sb..csize" */
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_add_at_pre_s_135(ArrayList sa, ArrayList sb, int i1, int i2,
         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", "sa..msize",
         "sb..msize"
4196     ensures "True" */
4197 {
4198     /*: 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)" */
4199     /*: assume "0 <= i1 & i1 < sa..csize" */
4200     sa.remove_at(i1);
4201     /*: assume "0 <= i2 & i2 <= sa..csize" */

```

```

4202     sa.add_at(i2, v2);
4203
4204     sb.add_at(i2, v2);
4205     sb.remove_at(i1);
4206
4207     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
4208 }
4209
4210 static void remove_at_add_at_between_s_136(ArrayList sa, ArrayList sb, int i1, int
4211     i2, Object v2)
4212 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4213     sa..contents = sb..contents & sa..csize = sb..csize"
4214     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
4215     "sb..msize"
4216     ensures "True" */
4217 {
4218     /*: assume "0 <= i1 & i1 < sa..csize" */
4219     sa.remove_at(i1);
4220     /*: assume "(i1 < i2 & (i2 - 1, v2) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
4221     sa..csize) | (i1 = i2 & (i1, v2) : sa..(old contents) & 0 <= i1 & i1 <
4222     sa..(old csize)) | (i1 > i2 & (ALL v. ((i1 - 1, v) : sa..contents) = ((i1,
4223     v) : sa..(old contents))) & 0 <= i1 - 1 & i1 - 1 < sa..csize & 0 <= i1 & i1
4224     < sa..(old csize))" */
4225     /*: assume "0 <= i2 & i2 <= sa..csize" */
4226     sa.add_at(i2, v2);
4227
4228     sb.add_at(i2, v2);
4229     sb.remove_at(i1);
4230
4231     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
4232 }
4233
4234 static void remove_at_add_at_post_s_137(ArrayList sa, ArrayList sb, int i1, int i2,
4235     Object v2)
4236 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4237     sa..contents = sb..contents & sa..csize = sb..csize"
4238     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
4239     "sb..msize"
4240     ensures "True" */
4241 {
4242     /*: assume "0 <= i1 & i1 < sa..csize" */
4243     sa.remove_at(i1);
4244     /*: assume "0 <= i2 & i2 <= sa..csize" */
4245     sa.add_at(i2, v2);
4246     /*: assume "(i1 < i2 & (i2 - 1, v2) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
4247     sa..csize) | (i1 = i2 & (i1, v2) : sa..(old contents) & 0 <= i1 & i1 <
4248     sa..(old csize)) | (i1 > i2 & (ALL v. ((i1, v) : sa..contents) = ((i1, v) :
4249     sa..(old contents))) & 0 <= i1 & i1 < sa..csize & 0 <= i1 & i1 < sa..(old
4250     csize))" */
4251     sb.add_at(i2, v2);
4252     sb.remove_at(i1);
4253
4254     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
4255 }
4256
4257 static void remove_at_add_at_post_c_137(ArrayList sa, ArrayList sb, int i1, int i2,
4258     Object v2)
4259 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4260     sa..contents = sb..contents & sa..csize = sb..csize"
4261     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
4262     "sb..msize"
4263     ensures "True" */
4264 {
4265     /*: assume "0 <= i1 & i1 < sa..csize" */

```



```

4253     sa.remove_at(i1);
4254     /*: assume "0 <= i2 & i2 <= sa..csize" */
4255     sa.add_at(i2, v2);
4256     /*: 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)))" */

4257     /*: assume "0 <= i2 & i2 <= sb..csize" */
4258     sb.add_at(i2, v2);
4259     /*: assume "0 <= i1 & i1 < sb..csize" */
4260     sb.remove_at(i1);
4261
4262     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
4263 }
4264
4265
4266 static void remove_at_get_pre_s_138(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 "(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" */
4273     /*: assume "0 <= i1 & i1 < sa..csize" */
4274     sa.remove_at(i1);
4275     /*: assume "0 <= i2 & i2 < sa..csize" */
4276     Object r2a = sa.get(i2);
4277
4278     Object r2b = sb.get(i2);
4279     sb.remove_at(i1);
4280
4281     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
4282 }
4283
4284 static void remove_at_get_pre_c_138(ArrayList sa, ArrayList sb, int i1, int i2)
4285 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4286     sa..contents = sb..contents & sa..csize = sb..csize"
4287 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4288 ensures "True" */
4289 {
4290     /*: 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)" */
4291     /*: assume "0 <= i1 & i1 < sa..csize" */
4292     sa.remove_at(i1);
4293     /*: assume "0 <= i2 & i2 < sa..csize" */
4294     Object r2a = sa.get(i2);
4295
4296     /*: assume "0 <= i2 & i2 < sb..csize" */
4297     Object r2b = sb.get(i2);
4298     /*: assume "0 <= i1 & i1 < sb..csize" */
4299     sb.remove_at(i1);
4300
4301     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
4302 }
4303
4304 static void remove_at_get_between_s_139(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" */
4311      sa.remove_at(i1);
4312      /*: 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" */
4313      /*: assume "0 <= i2 & i2 < sa..csize" */
4314      Object r2a = sa.get(i2);
4315
4316      Object r2b = sb.get(i2);
4317      sb.remove_at(i1);
4318
4319      /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
4320  }
4321
4322  static void remove_at_get_between_c_139(ArrayList sa, ArrayList sb, int i1, int i2)
4323  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
          sa..contents = sb..contents & sa..csize = sb..csize"
4324  modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4325  ensures "True" */
4326  {
4327      /*: assume "0 <= i1 & i1 < sa..csize" */
4328      sa.remove_at(i1);
4329      /*: 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)" */
4331      /*: assume "0 <= i2 & i2 < sa..csize" */
4332      Object r2a = sa.get(i2);
4333
4334      /*: assume "0 <= i2 & i2 < sb..csize" */
4335      Object r2b = sb.get(i2);
4336      /*: assume "0 <= i1 & i1 < sb..csize" */
4337      sb.remove_at(i1);
4338
4339      /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
          */
4340  }
4341
4342  static void remove_at_get_post_s_140(ArrayList sa, ArrayList sb, int i1, int i2)
4343  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
          sa..contents = sb..contents & sa..csize = sb..csize"
4344  modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4345  ensures "True" */
4346  {
4347      /*: assume "0 <= i1 & i1 < sa..csize" */
4348      sa.remove_at(i1);
4349      /*: assume "0 <= i2 & i2 < sa..csize" */
4350      Object r2a = sa.get(i2);
4351      /*: 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" */
4353
4354      Object r2b = sb.get(i2);
4355      sb.remove_at(i1);
4356
4357      /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
4358  }

```

```

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 static void remove_at_lastIndexOf_between_c_145(ArrayList sa, ArrayList sb, int i1,
4415 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,
4424     v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | ((EX i. (i,
4425     v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) : sa..contents & i1
4426     <= i & i < sa..csize) & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 <
4427     sa..(old csize)))" */
4428   int r2a = sa.lastIndexOf(v2);
4429
4430   int r2b = sb.lastIndexOf(v2);
4431   /*: assume "0 <= i1 & i1 < sb..csize" */
4432   sb.remove_at(i1);
4433
4434   /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
4435     */
4436 }
4437
4438 static void remove_at_lastIndexOf_post_c_146(ArrayList sa, ArrayList sb, int i1,
4439 Object v2)
4440 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4441          sa..contents = sb..contents & sa..csize = sb..csize"
4442 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4443 ensures "True" */
4444 {
4445   /*: assume "0 <= i1 & i1 < sa..csize" */
4446   sa.remove_at(i1);
4447   int r2a = sa.lastIndexOf(v2);
4448   /*: assume "~((r2a < 0 & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 <
4449     sa..(old csize)) | (0 <= r2a & r2a < i1 & (i1, v2) ~: sa..(old contents) & 0
4450     <= i1 & i1 < sa..(old csize)))" */
4451
4452   int r2b = sb.lastIndexOf(v2);
4453   /*: assume "0 <= i1 & i1 < sb..csize" */
4454   sb.remove_at(i1);
4455
4456   /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
4457     */
4458 }
4459
4460 static void remove_at_remove_at_pre_s_147(ArrayList sa, ArrayList sb, int i1, int
4461 i2)
4462 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4463          sa..contents = sb..contents & sa..csize = sb..csize"
4464 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4465 ensures "True" */
4466 {
4467   /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
4468     sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
4469     sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :
4470     sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
4471     sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
4472     sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
4473     <= i1 + 1 & i1 + 1 < sa..csize)" */
4474   /*: assume "0 <= i1 & i1 < sa..csize" */
4475   sa.remove_at(i1);
4476   /*: assume "0 <= i2 & i2 < sa..csize" */
4477   Object r2a = sa.remove_at(i2);

```

```

4462     Object r2b = sb.remove_at(i2);
4463     sb.remove_at(i1);
4464
4465     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
4466 }
4467
4468
4469 static void remove_at_remove_at_between_s_148(ArrayList sa, ArrayList sb, int i1,
4470     int i2)
4471 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4472     sa..contents = sb..contents & sa..csize = sb..csize"
4473     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4474     ensures "True" */
4475 {
4476     /*: assume "0 <= i1 & i1 < sa..csize" */
4477     sa.remove_at(i1);
4478     /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
4479     sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
4480     sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..(old contents)) = ((i2, v) :
4481     sa..contents)) & 0 <= i1 & i1 < sa..(old csize) & 0 <= i2 & i2 < sa..csize)
4482     | (sa..csize > i1 & i1 > i2 & (ALL v. ((i1, v) : sa..(old contents)) = ((i1,
4483     v) : sa..contents)) & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 & i1 <
4484     sa..csize)" */
4485     /*: assume "0 <= i2 & i2 < sa..csize" */
4486     Object r2a = sa.remove_at(i2);
4487
4488     Object r2b = sb.remove_at(i2);
4489     sb.remove_at(i1);
4490
4491     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
4492 }
4493
4494 static void remove_at_remove_at_post_s_149(ArrayList sa, ArrayList sb, int i1, int
4495     i2)
4496 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4497     sa..contents = sb..contents & sa..csize = sb..csize"
4498     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4499     ensures "True" */
4500 {
4501     /*: assume "0 <= i1 & i1 < sa..csize" */
4502     sa.remove_at(i1);
4503     /*: assume "0 <= i2 & i2 < sa..csize" */
4504     Object r2a = sa.remove_at(i2);
4505     /*: assume "(i1 < i2 & (i2 - 1, r2a) : sa..contents & 0 <= i2 - 1 & i2 - 1 <
4506     sa..csize) | (i1 = i2 & (i1, r2a) : sa..(old contents) & 0 <= i1 & i1 <
4507     sa..(old csize)) | (sa..csize + 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
4508     sa..(old contents)) = ((i1 - 1, v) : sa..contents)) & 0 <= i1 & i1 <
4509     sa..(old csize) & 0 <= i1 - 1 & i1 - 1 < sa..csize)" */
4510
4511     Object r2b = sb.remove_at(i2);
4512     sb.remove_at(i1);
4513
4514     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
4515 }
4516
4517 static void remove_at_remove_at_post_c_149(ArrayList sa, ArrayList sb, int i1, int
4518     i2)
4519 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4520     sa..contents = sb..contents & sa..csize = sb..csize"
4521     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4522     ensures "True" */
4523 {
4524     /*: assume "0 <= i1 & i1 < sa..csize" */
4525     sa.remove_at(i1);
4526     /*: assume "0 <= i2 & i2 < sa..csize" */

```

```

4514 Object r2a = sa.remove_at(i2);
4515 /*: 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))" */

4516
4517 /*: assume "0 <= i2 & i2 < sb..csize" */
4518 Object r2b = sb.remove_at(i2);
4519 /*: assume "0 <= i1 & i1 < sb..csize" */
4520 sb.remove_at(i1);
4521
4522 /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
*/
}

4523
4524 static void remove_at_remove_at_pre_s_150(ArrayList sa, ArrayList sb, int i1, int
i2)
4525
4526 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
sa..contents = sb..contents & sa..csize = sb..csize"
4527
4528 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4529 ensures "True" */
4530 {
4531 /*: 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)" */
4532 /*: assume "0 <= i1 & i1 < sa..csize" */
4533 sa.remove_at(i1);
4534 /*: assume "0 <= i2 & i2 < sa..csize" */
4535 sa.remove_at(i2);
4536
4537 sb.remove_at(i2);
4538 sb.remove_at(i1);
4539
4540 /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
4541 }
4542
4543 static void remove_at_remove_at_between_s_151(ArrayList sa, ArrayList sb, int i1,
int i2)
4544
4545 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
sa..contents = sb..contents & sa..csize = sb..csize"
4546
4547 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4548 ensures "True" */
4549 {
4550 /*: assume "0 <= i1 & i1 < sa..csize" */
4551 sa.remove_at(i1);
4552 /*: 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)" */
4553 /*: assume "0 <= i2 & i2 < sa..csize" */
4554 sa.remove_at(i2);
4555
4556 sb.remove_at(i2);
4557 sb.remove_at(i1);
4558
4559 /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
4560 }
4561
4562 static void remove_at_remove_at_post_s_152(ArrayList sa, ArrayList sb, int i1, int
i2)
4563
4564 /*: 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 &
         sa..contents = sb..contents & sa..csize = sb..csize"
4583 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4584 ensures "True" */
4585 {
4586     /*: assume "0 <= i1 & i1 < sa..csize" */
4587     sa.remove_at(i1);
4588     /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
4589     /*: ghost specvar sa__csize :: "int" = "sa..csize" */
4590     /*: assume "0 <= i2 & i2 < sa..csize" */
4591     sa.remove_at(i2);
4592     /*: 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))" */
4593
4594     /*: assume "0 <= i2 & i2 < sb..csize" */
4595     sb.remove_at(i2);
4596     /*: assume "0 <= i1 & i1 < sb..csize" */
4597     sb.remove_at(i1);
4598
4599     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
4600 }
4601
4602 static void remove_at_set_pre_s_153(ArrayList sa, ArrayList sb, int i1, int i2,
         Object v2)
4603 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
         sa..contents = sb..contents & sa..csize = sb..csize"
4604 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4605 ensures "True" */
4606 {
4607     /*: 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)" */
4608     /*: assume "0 <= i1 & i1 < sa..csize" */
4609     sa.remove_at(i1);
4610     /*: assume "0 <= i2 & i2 < sa..csize" */
4611
4612

```

```

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,
4622     Object v2)
4623 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4624     sa..contents = sb..contents & sa..csize = sb..csize"
4625 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4626 ensures "True" */
4627 {
4628     /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
4629     sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) : sa..contents & 0
4630     <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & (ALL
4631     v. ((i1, v) : sa..contents) = ((i2 + 1, v) : sa..contents)) & (i1, v2) :
4632     sa..contents & (i2 + 1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <=
4633     i2 + 1 & i2 + 1 < sa..csize) | (i1 > i2))" */
4634     /*: assume "0 <= i1 & i1 < sa..csize" */
4635     sa.remove_at(i1);
4636     /*: assume "0 <= i2 & i2 < sa..csize" */
4637     Object r2a = sa.set(i2, v2);
4638
4639     /*: assume "0 <= i2 & i2 < sb..csize" */
4640     Object r2b = sb.set(i2, v2);
4641     /*: assume "0 <= i1 & i1 < sb..csize" */
4642     sb.remove_at(i1);
4643
4644     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
4645     */
4646 }
4647
4648 static void remove_at_set_between_s_154(ArrayList sa, ArrayList sb, int i1, int i2,
4649     Object v2)
4650 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4651     sa..contents = sb..contents & sa..csize = sb..csize"
4652 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4653 ensures "True" */
4654 {
4655     /*: assume "0 <= i1 & i1 < sa..csize" */
4656     sa.remove_at(i1);
4657     /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
4658     sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa..contents & 0
4659     <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize) | (i1 = i2 & (ALL
4660     v. ((i1, v) : sa..(old contents)) = ((i2, v) : sa..contents)) & (i1, v2) :
4661     sa..(old contents) & (i2, v2) : sa..contents & 0 <= i1 & i1 < sa..(old
4662     csize) & 0 <= i2 & i2 < sa..csize) | i1 > i2" */
4663     /*: assume "0 <= i2 & i2 < sa..csize" */
4664     Object r2a = sa.set(i2, v2);
4665
4666     Object r2b = sb.set(i2, v2);
4667     sb.remove_at(i1);
4668
4669     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
4670 }
4671
4672 static void remove_at_set_between_c_154(ArrayList sa, ArrayList sb, int i1, int i2,
4673     Object v2)
4674 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4675     sa..contents = sb..contents & sa..csize = sb..csize"
4676 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4677 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 &
         sa..contents = sb..contents & sa..csize = sb..csize"
4681     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4682     ensures "True" */
4683 {
4684     /*: assume "0 <= i1 & i1 < sa..csize" */
4685     sa.remove_at(i1);
4686     /*: assume "0 <= i2 & i2 < sa..csize" */
4687     Object r2a = sa.set(i2, v2);
4688     /*: 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" */
4689
4690     Object r2b = sb.set(i2, v2);
4691     sb.remove_at(i1);
4692
4693     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
4694 }
4695
4696 static void remove_at_set_post_c_155(ArrayList sa, ArrayList sb, int i1, int i2,
         Object v2)
4697 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
         sa..contents = sb..contents & sa..csize = sb..csize"
4698     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4699     ensures "True" */
4700 {
4701     /*: assume "0 <= i1 & i1 < sa..csize" */
4702     sa.remove_at(i1);
4703     /*: assume "0 <= i2 & i2 < sa..csize" */
4704     Object r2a = sa.set(i2, v2);
4705     /*: 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)" */
4706
4707     /*: assume "0 <= i2 & i2 < sb..csize" */
4708     Object r2b = sb.set(i2, v2);
4709     /*: assume "0 <= i1 & i1 < sb..csize" */
4710     sb.remove_at(i1);
4711
4712
4713

```

```

4714     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
4715         */
4716 }
4717
4718 static void remove_at_set_pre_s_156(ArrayList sa, ArrayList sb, int i1, int i2,
4719     Object v2)
4720 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4721     sa..contents = sb..contents & sa..csize = sb..csize"
4722     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4723     ensures "True" */
4724 {
4725     /*: assume "(i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
4726     sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) : sa..contents & 0
4727     <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & (i2
4728     + 1, v2) : sa..contents & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 > i2)" */
4729     /*: assume "0 <= i1 & i1 < sa..csize" */
4730     sa.remove_at(i1);
4731     /*: assume "0 <= i2 & i2 < sa..csize" */
4732     sa.set(i2, v2);
4733
4734     sb.set(i2, v2);
4735     sb.remove_at(i1);
4736
4737     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
4738 }
4739
4740 static void remove_at_set_between_s_157(ArrayList sa, ArrayList sb, int i1, int i2,
4741     Object v2)
4742 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4743     sa..contents = sb..contents & sa..csize = sb..csize"
4744     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4745     ensures "True" */
4746 {
4747     /*: assume "0 <= i1 & i1 < sa..csize" */
4748     sa.remove_at(i1);
4749     /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
4750     sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa..contents & 0
4751     <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize) | (i1 = i2 & (i2,
4752     v2) : sa..contents & 0 <= i2 & i2 < sa..csize) | i1 > i2" */
4753     /*: assume "0 <= i2 & i2 < sa..csize" */
4754     sa.set(i2, v2);
4755
4756     sb.set(i2, v2);
4757     sb.remove_at(i1);
4758
4759     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
4760 }
4761
4762 static void remove_at_set_between_c_157(ArrayList sa, ArrayList sb, int i1, int i2,
4763     Object v2)
4764 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4765     sa..contents = sb..contents & sa..csize = sb..csize"
4766     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4767     ensures "True" */
4768 {
4769     /*: assume "0 <= i1 & i1 < sa..csize" */
4770     sa.remove_at(i1);
4771     /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
4772     sa..contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa..contents & 0
4773     <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa..csize) | (i1 = i2 & (i2,
4774     v2) : sa..contents & 0 <= i2 & i2 < sa..csize) | i1 > i2)" */
4775     /*: assume "0 <= i2 & i2 < sa..csize" */
4776     sa.set(i2, v2);
4777
4778     /*: assume "0 <= i2 & i2 < sb..csize" */

```

```

4766     sb.set(i2, v2);
4767     /*: assume "0 <= i1 & i1 < sb..csize" */
4768     sb.remove_at(i1);
4769
4770     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
4771 }
4772
4773 static void remove_at_set_post_s_158(ArrayList sa, ArrayList sb, int i1, int i2,
4774     Object v2)
4775 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4776     sa..contents = sb..contents & sa..csize = sb..csize"
4777 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4778 ensures "True" */
4779 {
4780     /*: assume "0 <= i1 & i1 < sa..csize" */
4781     sa.remove_at(i1);
4782     /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
4783     /*: ghost specvar sa__csize :: "int" = "sa..csize" */
4784     /*: assume "0 <= i2 & i2 < sa..csize" */
4785     sa.set(i2, v2);
4786     /*: assume "(i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
4787     sa__contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa__contents & 0
4788     <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa__csize) | (i1 = i2 & (i2,
4789     v2) : sa__contents & 0 <= i2 & i2 < sa__csize) | i1 > i2" */
4790
4791     sb.set(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_set_post_c_158(ArrayList sa, ArrayList sb, int i1, int i2,
4798     Object v2)
4799 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4800     sa..contents = sb..contents & sa..csize = sb..csize"
4801 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4802 ensures "True" */
4803 {
4804     /*: assume "0 <= i1 & i1 < sa..csize" */
4805     sa.remove_at(i1);
4806     /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
4807     /*: ghost specvar sa__csize :: "int" = "sa..csize" */
4808     /*: assume "0 <= i2 & i2 < sa..csize" */
4809     sa.set(i2, v2);
4810     /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
4811     sa__contents)) & (i2 - 1, v2) : sa..contents & (i2, v2) : sa__contents & 0
4812     <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 < sa__csize) | (i1 = i2 & (i2,
4813     v2) : sa__contents & 0 <= i2 & i2 < sa__csize) | i1 > i2)" */
4814
4815     /*: assume "0 <= i2 & i2 < sb..csize" */
4816     sb.set(i2, v2);
4817     /*: assume "0 <= i1 & i1 < sb..csize" */
4818     sb.remove_at(i1);
4819
4820     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
4821 }
4822
4823 static void remove_at_size_pre_s_159(ArrayList sa, ArrayList sb, int i1)
4824 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4825     sa..contents = sb..contents & sa..csize = sb..csize"
4826 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4827 ensures "True" */
4828 {
4829     /*: assume "False" */
4830     /*: assume "0 <= i1 & i1 < sa..csize" */

```

```

4823     sa.remove_at(i1);
4824     int r2a = sa.size();
4825
4826     int r2b = sb.size();
4827     sb.remove_at(i1);
4828
4829     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
4830 }
4831
4832 static void remove_at_size_pre_c_159(ArrayList sa, ArrayList sb, int i1)
4833 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4834     sa..contents = sb..contents & sa..csize = sb..csize"
4835     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4836     ensures "True" */
4837 {
4838     /*: assume "~(False)" */
4839     /*: assume "0 <= i1 & i1 < sa..csize" */
4840     sa.remove_at(i1);
4841     int r2a = sa.size();
4842
4843     int r2b = sb.size();
4844     /*: assume "0 <= i1 & i1 < sb..csize" */
4845     sb.remove_at(i1);
4846
4847     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
4848         */
4849 }
4850
4851 static void remove_at_size_between_s_160(ArrayList sa, ArrayList sb, int i1)
4852 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4853     sa..contents = sb..contents & sa..csize = sb..csize"
4854     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4855     ensures "True" */
4856 {
4857     /*: assume "0 <= i1 & i1 < sa..csize" */
4858     sa.remove_at(i1);
4859     /*: assume "False" */
4860     int r2a = sa.size();
4861
4862     int r2b = sb.size();
4863     sb.remove_at(i1);
4864
4865     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
4866 }
4867
4868 static void remove_at_size_between_c_160(ArrayList sa, ArrayList sb, int i1)
4869 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4870     sa..contents = sb..contents & sa..csize = sb..csize"
4871     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
4872     ensures "True" */
4873 {
4874     /*: assume "0 <= i1 & i1 < sa..csize" */
4875     sa.remove_at(i1);
4876     /*: assume "~(False)" */
4877     int r2a = sa.size();
4878
4879     int r2b = sb.size();
4880     /*: assume "0 <= i1 & i1 < sb..csize" */
4881     sb.remove_at(i1);
4882
4883     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
4884         */
4885 }
4886
4887 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
4921  static void set_add_at_pre_s_162(ArrayList sa, ArrayList sb, int i1, Object v1, int
4922  i2, Object v2)
4923  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4924          sa..contents = sb..contents & sa..csize = sb..csize"
4925  modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
4926          "sb..msize"
4927  ensures "True" */
4928  {
4929      /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
4930          sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | (i1 > i2 & (ALL v. ((i1
4931          - 1, v) : sa..contents) = ((i1, v) : sa..contents)) & (i1 - 1, v1) :
4932          sa..contents & (i1, v1) : sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize &
4933          0 <= i1 & i1 < sa..csize)" */
4934      /*: assume "0 <= i1 & i1 < sa..csize" */
4935      Object r1a = sa.set(i1, v1);
4936      /*: assume "0 <= i2 & i2 <= sa..csize" */
4937      sa.add_at(i2, v2);
4938
4939      sb.add_at(i2, v2);
4940      Object r1b = sb.set(i1, v1);
4941
4942      /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
4943  }
4944
4945  static void set_add_at_pre_c_162(ArrayList sa, ArrayList sb, int i1, Object v1, int
4946  i2, Object v2)
4947  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4948          sa..contents = sb..contents & sa..csize = sb..csize"
4949  modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
4950          "sb..msize"

```

```

4942     ensures "True" */
4943 {
4944     /*: 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))" */
4945     /*: assume "0 <= i1 & i1 < sa..csize" */
4946     Object r1a = sa.set(i1, v1);
4947     /*: assume "0 <= i2 & i2 <= sa..csize" */
4948     sa.add_at(i2, v2);
4949
4950     /*: assume "0 <= i2 & i2 <= sb..csize" */
4951     sb.add_at(i2, v2);
4952     /*: assume "0 <= i1 & i1 < sb..csize" */
4953     Object r1b = sb.set(i1, v1);
4954
4955     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
*/
4956 }
4957
4958 static void set_add_at_between_s_163(ArrayList sa, ArrayList sb, int i1, Object v1,
int i2, Object v2)
4959 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4960 sa..contents = sb..contents & sa..csize = sb..csize"
4961 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
"sb..msize"
4962 ensures "True" */
4963 {
4964     /*: assume "0 <= i1 & i1 < sa..csize" */
4965     Object r1a = sa.set(i1, v1);
4966     /*: 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)" */
4967     /*: assume "0 <= i2 & i2 <= sa..csize" */
4968     sa.add_at(i2, v2);
4969
4970     sb.add_at(i2, v2);
4971     Object r1b = sb.set(i1, v1);
4972
4973     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
4974 }
4975
4976 static void set_add_at_between_c_163(ArrayList sa, ArrayList sb, int i1, Object v1,
int i2, Object v2)
4977 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
4978 sa..contents = sb..contents & sa..csize = sb..csize"
4979 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
"sb..msize"
4980 ensures "True" */
4981 {
4982     /*: assume "0 <= i1 & i1 < sa..csize" */
4983     Object r1a = sa.set(i1, v1);
4984     /*: 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))" */
4985     /*: assume "0 <= i2 & i2 <= sa..csize" */
4986     sa.add_at(i2, v2);
4987
4988     /*: assume "0 <= i2 & i2 <= sb..csize" */
4989     sb.add_at(i2, v2);
4990     /*: assume "0 <= i1 & i1 < sb..csize" */
4991     Object r1b = sb.set(i1, v1);
4992

```

```

4993     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
4994         */
4995 }
4996
4997 static void set_add_at_post_s_164(ArrayList sa, ArrayList sb, int i1, Object v1,
4998     int i2, Object v2)
4999 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5000     sa..contents = sb..contents & sa..csize = sb..csize"
5001     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
5002     "sb..msize"
5003     ensures "True" */
5004 {
5005     /*: assume "0 <= i1 & i1 < sa..csize" */
5006     Object r1a = sa.set(i1, v1);
5007     /*: assume "0 <= i2 & i2 <= sa..csize" */
5008     sa.add_at(i2, v2);
5009     /*: assume "i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | (i1 > i2 &
5010     (i1, r1a) : sa..contents & (i1, v1) : sa..contents & r1a = v1 & 0 <= i1 & i1
5011     < sa..csize)" */
5012
5013     sb.add_at(i2, v2);
5014     Object r1b = sb.set(i1, v1);
5015
5016     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
5017 }
5018
5019 static void set_add_at_post_c_164(ArrayList sa, ArrayList sb, int i1, Object v1,
5020     int i2, Object v2)
5021 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5022     sa..contents = sb..contents & sa..csize = sb..csize"
5023     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
5024     "sb..msize"
5025     ensures "True" */
5026 {
5027     /*: assume "0 <= i1 & i1 < sa..csize" */
5028     Object r1a = sa.set(i1, v1);
5029     /*: assume "0 <= i2 & i2 <= sa..csize" */
5030     sa.add_at(i2, v2);
5031     /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | (i1 > i2 &
5032     (i1, r1a) : sa..contents & (i1, v1) : sa..contents & r1a = v1 & 0 <= i1 & i1
5033     < sa..csize))" */
5034
5035     /*: assume "0 <= i2 & i2 <= sb..csize" */
5036     sb.add_at(i2, v2);
5037     /*: assume "0 <= i1 & i1 < sb..csize" */
5038     Object r1b = sb.set(i1, v1);
5039
5040     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
5041         */
5042 }
5043
5044 static void set_get_pre_s_165(ArrayList sa, ArrayList sb, int i1, Object v1, int i2)
5045 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5046     sa..contents = sb..contents & sa..csize = sb..csize"
5047     modifies "sa..contents", "sb..contents"
5048     ensures "True" */
5049 {
5050     /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & 0 <= i1 & i1 <
5051     sa..csize) | i1 > i2" */
5052     /*: assume "0 <= i1 & i1 < sa..csize" */
5053     Object r1a = sa.set(i1, v1);
5054     /*: assume "0 <= i2 & i2 < sa..csize" */
5055     Object r2a = sa.get(i2);
5056
5057     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 &
        sa..contents = sb..contents & sa..csize = sb..csize"
5054     modifies "sa..contents", "sb..contents"
5055     ensures "True" */
5056 {
5057     /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & 0 <= i1 & i1 <
        sa..csize) | i1 > i2)" */
5058     /*: assume "0 <= i1 & i1 < sa..csize" */
5059     Object r1a = sa.set(i1, v1);
5060     /*: assume "0 <= i2 & i2 < sa..csize" */
5061     Object r2a = sa.get(i2);
5062
5063     /*: assume "0 <= i2 & i2 < sb..csize" */
5064     Object r2b = sb.get(i2);
5065     /*: assume "0 <= i1 & i1 < sb..csize" */
5066     Object r1b = sb.set(i1, v1);
5067
5068     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
5069 }
5070
5071 static void set_get_between_s_166(ArrayList sa, ArrayList sb, int i1, Object v1,
5072     int i2)
5073 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
5074     modifies "sa..contents", "sb..contents"
5075     ensures "True" */
5076 {
5077     /*: assume "0 <= i1 & i1 < sa..csize" */
5078     Object r1a = sa.set(i1, v1);
5079     /*: assume "i1 < i2 | (i1 = i2 & r1a = v1) | i1 > i2" */
5080     /*: assume "0 <= i2 & i2 < sa..csize" */
5081     Object r2a = sa.get(i2);
5082
5083     Object r2b = sb.get(i2);
5084     Object r1b = sb.set(i1, v1);
5085
5086     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
5087 }
5088
5089 static void set_get_between_c_166(ArrayList sa, ArrayList sb, int i1, Object v1,
5090     int i2)
5091 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
5092     modifies "sa..contents", "sb..contents"
5093     ensures "True" */
5094 {
5095     /*: assume "0 <= i1 & i1 < sa..csize" */
5096     Object r1a = sa.set(i1, v1);
5097     /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v1) | i1 > i2)" */
5098     /*: assume "0 <= i2 & i2 < sa..csize" */
5099     Object r2a = sa.get(i2);
5100
5101     /*: assume "0 <= i2 & i2 < sb..csize" */
5102     Object r2b = sb.get(i2);
5103     /*: assume "0 <= i1 & i1 < sb..csize" */
5104     Object r1b = sb.set(i1, v1);
5105

```



```

5106     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
5107         sb..csize)" */
5108 }
5109
5110 static void set_get_post_s_167(ArrayList sa, ArrayList sb, int i1, Object v1, int
5111     i2)
5112 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5113     sa..contents = sb..contents & sa..csize = sb..csize"
5114     modifies "sa..contents", "sb..contents"
5115     ensures "True" */
5116 {
5117     /*: assume "0 <= i1 & i1 < sa..csize" */
5118     Object r1a = sa.set(i1, v1);
5119     /*: assume "0 <= i2 & i2 < sa..csize" */
5120     Object r2a = sa.get(i2);
5121     /*: assume "i1 < i2 | (i1 = i2 & r1a = v1) | i1 > i2" */
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 =
5126         sb..csize" */
5127 }
5128
5129 static void set_get_post_c_167(ArrayList sa, ArrayList sb, int i1, Object v1, int
5130     i2)
5131 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5132     sa..contents = sb..contents & sa..csize = sb..csize"
5133     modifies "sa..contents", "sb..contents"
5134     ensures "True" */
5135 {
5136     /*: assume "0 <= i1 & i1 < sa..csize" */
5137     Object r1a = sa.set(i1, v1);
5138     /*: assume "0 <= i2 & i2 < sa..csize" */
5139     Object r2a = sa.get(i2);
5140     /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v1) | i1 > i2)" */
5141     /*: assume "0 <= i2 & i2 < sb..csize" */
5142     Object r2b = sb.get(i2);
5143     /*: assume "0 <= i1 & i1 < sb..csize" */
5144     Object r1b = sb.set(i1, v1);
5145
5146     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
5147         sb..csize)" */
5148 }
5149
5150 static void set_indexOf_pre_s_168(ArrayList sa, ArrayList sb, int i1, Object v1,
5151     Object v2)
5152 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5153     sa..contents = sb..contents & sa..csize = sb..csize"
5154     modifies "sa..contents", "sb..contents"
5155     ensures "True" */
5156 {
5157     /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
5158         v2) | (EX i. (i, v2) : sa..contents & 0 <= i & i < i1) | ~(EX i. (i, v2) :
5159         sa..contents & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 <
5160         sa..csize & v1 = v2) | ~(EX i. (i, v2) : sa..contents & 0 <= i & i <= i1) &
5161         (EX i. (i, v2) : sa..contents & i1 < i & i < sa..csize) & v1 ~= v2)" */
5162     /*: assume "0 <= i1 & i1 < sa..csize" */
5163     Object r1a = sa.set(i1, v1);
5164     int r2a = sa.indexOf(v2);
5165
5166     int r2b = sb.indexOf(v2);
5167     Object r1b = sb.set(i1, v1);

```

```

5161     /*: assert "ria = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
5162         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 "~(ria = 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 "ria = 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 <

```

```

5209         sa..csize & r1a = v2) | ~(EX i. (i, v2) : sa..contents & 0 <= i & i <= i1)
5210         & (EX i. (i, v2) : sa..contents & i1 < i & i < sa..csize) & r1a ~= v2))" */
5211     int r2a = sa.index0f(v2);
5212
5213     int r2b = sb.index0f(v2);
5214     /*: assume "0 <= i1 & i1 < sb..csize" */
5215     Object r1b = sb.set(i1, v1);
5216
5217     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
5218     sb..csize)" */
5219 }
5220
5221 static void set_index0f_post_s_170(ArrayList sa, ArrayList sb, int i1, Object v1,
5222     Object v2)
5223 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5224     sa..contents = sb..contents & sa..csize = sb..csize"
5225     modifies "sa..contents", "sb..contents"
5226     ensures "True" */
5227 {
5228     /*: assume "0 <= i1 & i1 < sa..csize" */
5229     Object r1a = sa.set(i1, v1);
5230     int r2a = sa.index0f(v2);
5231     /*: assume "(r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1) | (r2a = i1 & r1a =
5232     v2) | (r2a > i1 & r1a ~= v2)" */
5233
5234     int r2b = sb.index0f(v2);
5235     Object r1b = sb.set(i1, v1);
5236
5237     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
5238     sb..csize" */
5239 }
5240
5241 static void set_index0f_post_c_170(ArrayList sa, ArrayList sb, int i1, Object v1,
5242     Object v2)
5243 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5244     sa..contents = sb..contents & sa..csize = sb..csize"
5245     modifies "sa..contents", "sb..contents"
5246     ensures "True" */
5247 {
5248     /*: assume "0 <= i1 & i1 < sa..csize" */
5249     Object r1a = sa.set(i1, v1);
5250     int r2a = sa.index0f(v2);
5251     /*: assume "~((r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1) | (r2a = i1 & r1a =
5252     v2) | (r2a > i1 & r1a ~= v2))" */
5253
5254     int r2b = sb.index0f(v2);
5255     /*: assume "0 <= i1 & i1 < sb..csize" */
5256     Object r1b = sb.set(i1, v1);
5257
5258     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
5259     sb..csize)" */
5260 }
5261
5262 static void set_lastIndex0f_pre_s_171(ArrayList sa, ArrayList sb, int i1, Object
5263     v1, Object v2)
5264 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5265     sa..contents = sb..contents & sa..csize = sb..csize"
5266     modifies "sa..contents", "sb..contents"
5267     ensures "True" */
5268 {
5269     /*: assume "(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
5270     v2) | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
5271     sa..contents & i1 <= i & i < sa..csize) & v1 ~= v2) | ((i1, v2) :
5272     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)" */
5260 /*: assume "0 <= i1 & i1 < sa..csize" */
5261 Object r1a = sa.set(i1, v1);
5262 int r2a = sa.lastIndexOf(v2);
5263
5264 int r2b = sb.lastIndexOf(v2);
5265 Object r1b = sb.set(i1, v1);
5266
5267 /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
5268 }
5269
5270 static void set_lastIndexOf_pre_c_171(ArrayList sa, ArrayList sb, int i1, Object
        v1, Object v2)
5271 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5272         sa..contents = sb..contents & sa..csize = sb..csize"
5273     modifies "sa..contents", "sb..contents"
5274     ensures "True" */
5275 {
5276     /*: 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)))" */
5277     /*: assume "0 <= i1 & i1 < sa..csize" */
5278     Object r1a = sa.set(i1, v1);
5279     int r2a = sa.lastIndexOf(v2);
5280
5281     int r2b = sb.lastIndexOf(v2);
5282     /*: assume "0 <= i1 & i1 < sb..csize" */
5283     Object r1b = sb.set(i1, v1);
5284
5285     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
5286 }
5287
5288 static void set_lastIndexOf_between_s_172(ArrayList sa, ArrayList sb, int i1,
        Object v1, Object v2)
5289 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5290         sa..contents = sb..contents & sa..csize = sb..csize"
5291     modifies "sa..contents", "sb..contents"
5292     ensures "True" */
5293 {
5294     /*: assume "0 <= i1 & i1 < sa..csize" */
5295     Object r1a = sa.set(i1, v1);
5296     /*: 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)" */
5297     int r2a = sa.lastIndexOf(v2);
5298
5299     int r2b = sb.lastIndexOf(v2);
5300     Object r1b = sb.set(i1, v1);
5301
5302     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
5303 }
5304
5305 static void set_lastIndexOf_between_c_172(ArrayList sa, ArrayList sb, int i1,
        Object v1, Object v2)
5306 /*: 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
5314         ~= v2) | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2)
5315         : sa..contents & i1 <= i & i < sa..csize) & r1a ~= v2) | ((i1, v2) :
5316         sa..contents & 0 <= i1 & i1 < sa..csize & ~(EX i. (i, v2) : sa..contents &
5317         i1 < i & i < sa..csize) & r1a = v2) | (EX i. (i, v2) : sa..contents & i1 < i
5318         & i < sa..csize))" */
5319     int r2a = sa.lastIndexOf(v2);
5320
5321     int r2b = sb.lastIndexOf(v2);
5322     /*: assume "0 <= i1 & i1 < sb..csize" */
5323     Object r1b = sb.set(i1, v1);
5324
5325     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
5326         sb..csize)" */
5327 }
5328
5329 static void set_lastIndexOf_post_s_173(ArrayList sa, ArrayList sb, int i1, Object
5330     v1, Object v2)
5331 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5332     sa..contents = sb..contents & sa..csize = sb..csize"
5333     modifies "sa..contents", "sb..contents"
5334     ensures "True" */
5335 {
5336     /*: assume "0 <= i1 & i1 < sa..csize" */
5337     Object r1a = sa.set(i1, v1);
5338     int r2a = sa.lastIndexOf(v2);
5339     /*: assume "(r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1 & r1a ~= v2) | (r2a =
5340         i1 & r1a = v2) | r2a > i1" */
5341
5342     int r2b = sb.lastIndexOf(v2);
5343     Object r1b = sb.set(i1, v1);
5344
5345     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
5346         sb..csize" */
5347 }
5348
5349 static void set_lastIndexOf_post_c_173(ArrayList sa, ArrayList sb, int i1, Object
5350     v1, Object v2)
5351 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5352     sa..contents = sb..contents & sa..csize = sb..csize"
5353     modifies "sa..contents", "sb..contents"
5354     ensures "True" */
5355 {
5356     /*: assume "0 <= i1 & i1 < sa..csize" */
5357     Object r1a = sa.set(i1, v1);
5358     int r2a = sa.lastIndexOf(v2);
5359     /*: assume "~((r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1 & r1a ~= v2) | (r2a
5360         = i1 & r1a = v2) | r2a > i1)" */
5361
5362     int r2b = sb.lastIndexOf(v2);
5363     /*: assume "0 <= i1 & i1 < sb..csize" */
5364     Object r1b = sb.set(i1, v1);
5365
5366     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
5367         sb..csize)" */
5368 }
5369
5370 static void set_remove_at_pre_s_174(ArrayList sa, ArrayList sb, int i1, Object v1,
5371     int i2)

```

```

5359  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5360          sa..contents = sb..contents & sa..csize = sb..csize"
5361  modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5362  ensures "True" */
5363  {
5364      /*: 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)" */
5365      /*: assume "0 <= i1 & i1 < sa..csize" */
5366      Object r1a = sa.set(i1, v1);
5367      /*: assume "0 <= i2 & i2 < sa..csize" */
5368      Object r2a = sa.remove_at(i2);
5369
5370      Object r2b = sb.remove_at(i2);
5371      Object r1b = sb.set(i1, v1);
5372
5373      /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
          sb..csize" */
5374  }
5375
5376  static void set_remove_at_pre_c_174(ArrayList sa, ArrayList sb, int i1, Object v1,
          int i2)
5377  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
          sa..contents = sb..contents & sa..csize = sb..csize"
5378  modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5379  ensures "True" */
5380  {
5381      /*: 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))" */
5383      /*: assume "0 <= i1 & i1 < sa..csize" */
5384      Object r1a = sa.set(i1, v1);
5385      /*: assume "0 <= i2 & i2 < sa..csize" */
5386      Object r2a = sa.remove_at(i2);
5387
5388      /*: assume "0 <= i2 & i2 < sb..csize" */
5389      Object r2b = sb.remove_at(i2);
5390      /*: assume "0 <= i1 & i1 < sb..csize" */
5391      Object r1b = sb.set(i1, v1);
5392
5393      /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
          sb..csize)" */
5394  }
5395
5396  static void set_remove_at_between_s_175(ArrayList sa, ArrayList sb, int i1, Object
          v1, int i2)
5397  /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
          sa..contents = sb..contents & sa..csize = sb..csize"
5398  modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5399  ensures "True" */
5400  {
5401      /*: assume "0 <= i1 & i1 < sa..csize" */
5402      Object r1a = sa.set(i1, v1);
5403      /*: 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"

```

```

5455     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5456     ensures "True" */
5457 {
5458     /*: assume "0 <= i1 & i1 < sa..csize" */
5459     Object r1a = sa.set(i1, v1);
5460     /*: assume "0 <= i2 & i2 < sa..csize" */
5461     Object r2a = sa.remove_at(i2);
5462     /*: 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))" */
5463
5464     /*: assume "0 <= i2 & i2 < sb..csize" */
5465     Object r2b = sb.remove_at(i2);
5466     /*: assume "0 <= i1 & i1 < sb..csize" */
5467     Object r1b = sb.set(i1, v1);
5468
5469     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
5470 }
5471
5472 static void set_remove_at_pre_s_177(ArrayList sa, ArrayList sb, int i1, Object v1,
    int i2)
5473 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5474     sa..contents = sb..contents & sa..csize = sb..csize"
5475     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5476     ensures "True" */
5477 {
5478     /*: 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)" */
5479     /*: assume "0 <= i1 & i1 < sa..csize" */
5480     Object r1a = sa.set(i1, v1);
5481     /*: assume "0 <= i2 & i2 < sa..csize" */
5482     sa.remove_at(i2);
5483
5484     sb.remove_at(i2);
5485     Object r1b = sb.set(i1, v1);
5486
5487     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
5488 }
5489
5490 static void set_remove_at_pre_c_177(ArrayList sa, ArrayList sb, int i1, Object v1,
    int i2)
5491 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5492     sa..contents = sb..contents & sa..csize = sb..csize"
5493     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5494     ensures "True" */
5495 {
5496     /*: 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))" */
5497     /*: assume "0 <= i1 & i1 < sa..csize" */
5498     Object r1a = sa.set(i1, v1);
5499     /*: assume "0 <= i2 & i2 < sa..csize" */
5500     sa.remove_at(i2);
5501

```



```

5502     /*: assume "0 <= i2 & i2 < sb..csize" */
5503     sb.remove_at(i2);
5504     /*: assume "0 <= i1 & i1 < sb..csize" */
5505     Object r1b = sb.set(i1, v1);
5506
5507     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
5508 }
5509
5510 static void set_remove_at_between_s_178(ArrayList sa, ArrayList sb, int i1, Object
    v1, int i2)
5511 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5512             sa..contents = sb..contents & sa..csize = sb..csize"
5513 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5514 ensures "True" */
5515 {
5516     /*: assume "0 <= i1 & i1 < sa..csize" */
5517     Object r1a = sa.set(i1, v1);
5518     /*: 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)" */
5519     /*: assume "0 <= i2 & i2 < sa..csize" */
5520     sa.remove_at(i2);
5521
5522     sb.remove_at(i2);
5523     Object r1b = sb.set(i1, v1);
5524
5525     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
5526 }
5527
5528 static void set_remove_at_between_c_178(ArrayList sa, ArrayList sb, int i1, Object
    v1, int i2)
5529 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5530             sa..contents = sb..contents & sa..csize = sb..csize"
5531 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5532 ensures "True" */
5533 {
5534     /*: assume "0 <= i1 & i1 < sa..csize" */
5535     Object r1a = sa.set(i1, v1);
5536     /*: 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)" */
5537     /*: assume "0 <= i2 & i2 < sa..csize" */
5538     sa.remove_at(i2);
5539
5540     /*: assume "0 <= i2 & i2 < sb..csize" */
5541     sb.remove_at(i2);
5542     /*: assume "0 <= i1 & i1 < sb..csize" */
5543     Object r1b = sb.set(i1, v1);
5544
5545     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
5546 }
5547
5548 static void set_remove_at_post_s_179(ArrayList sa, ArrayList sb, int i1, Object v1,
    int i2)
5549 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5550             sa..contents = sb..contents & sa..csize = sb..csize"
5551 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5552 ensures "True" */
5553 {

```

```

5554     /*: assume "0 <= i1 & i1 < sa..csize" */
5555     Object r1a = sa.set(i1, v1);
5556     /*: assume "0 <= i2 & i2 < sa..csize" */
5557     sa.remove_at(i2);
5558     /*: 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)" */

5559
5560     sb.remove_at(i2);
5561     Object r1b = sb.set(i1, v1);
5562
5563     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
5564 }
5565
5566 static void set_remove_at_post_c_179(ArrayList sa, ArrayList sb, int i1, Object v1,
    int i2)
5567 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5568     sa..contents = sb..contents & sa..csize = sb..csize"
5569     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
5570     ensures "True" */
5571 {
5572     /*: assume "0 <= i1 & i1 < sa..csize" */
5573     Object r1a = sa.set(i1, v1);
5574     /*: assume "0 <= i2 & i2 < sa..csize" */
5575     sa.remove_at(i2);
5576     /*: 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))" */

5577
5578     /*: assume "0 <= i2 & i2 < sb..csize" */
5579     sb.remove_at(i2);
5580     /*: assume "0 <= i1 & i1 < sb..csize" */
5581     Object r1b = sb.set(i1, v1);
5582
5583     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
5584 }
5585
5586 static void set_set_pre_s_180(ArrayList sa, ArrayList sb, int i1, Object v1, int
    i2, Object v2)
5587 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5588     sa..contents = sb..contents & sa..csize = sb..csize"
5589     modifies "sa..contents", "sb..contents"
5590     ensures "True" */
5591 {
5592     /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
        sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | i1 > i2" */
5593     /*: assume "0 <= i1 & i1 < sa..csize" */
5594     Object r1a = sa.set(i1, v1);
5595     /*: assume "0 <= i2 & i2 < sa..csize" */
5596     Object r2a = sa.set(i2, v2);
5597
5598     Object r2b = sb.set(i2, v2);
5599     Object r1b = sb.set(i1, v1);
5600
5601     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
5602 }
5603
5604 static void set_set_pre_c_180(ArrayList sa, ArrayList sb, int i1, Object v1, int
    i2, Object v2)
5605 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5606     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) :
5611         sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | i1 > i2)" */
5612     /*: assume "0 <= i1 & i1 < sa..csize" */
5613     Object r1a = sa.set(i1, v1);
5614     /*: assume "0 <= i2 & i2 < sa..csize" */
5615     Object r2a = sa.set(i2, v2);
5616
5617     /*: assume "0 <= i2 & i2 < sb..csize" */
5618     Object r2b = sb.set(i2, v2);
5619     /*: assume "0 <= i1 & i1 < sb..csize" */
5620     Object r1b = sb.set(i1, v1);
5621
5622     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
5623         sb..csize)" */
5624 }
5625
5626 static void set_set_between_s_181(ArrayList sa, ArrayList sb, int i1, Object v1,
5627     int i2, Object v2)
5628 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5629     sa..contents = sb..contents & sa..csize = sb..csize"
5630 modifies "sa..contents", "sb..contents"
5631 ensures "True" */
5632 {
5633     /*: assume "0 <= i1 & i1 < sa..csize" */
5634     Object r1a = sa.set(i1, v1);
5635     /*: assume "i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | i1 > i2" */
5636     /*: assume "0 <= i2 & i2 < sa..csize" */
5637     Object r2a = sa.set(i2, v2);
5638
5639     Object r2b = sb.set(i2, v2);
5640     Object r1b = sb.set(i1, v1);
5641
5642     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
5643         sb..csize" */
5644 }
5645
5646 static void set_set_between_c_181(ArrayList sa, ArrayList sb, int i1, Object v1,
5647     int i2, Object v2)
5648 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5649     sa..contents = sb..contents & sa..csize = sb..csize"
5650 modifies "sa..contents", "sb..contents"
5651 ensures "True" */
5652 {
5653     /*: assume "0 <= i1 & i1 < sa..csize" */
5654     Object r1a = sa.set(i1, v1);
5655     /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | i1 > i2)" */
5656     /*: assume "0 <= i2 & i2 < sa..csize" */
5657     Object r2a = sa.set(i2, v2);
5658
5659     /*: assume "0 <= i2 & i2 < sb..csize" */
5660     Object r2b = sb.set(i2, v2);
5661     /*: assume "0 <= i1 & i1 < sb..csize" */
5662     Object r1b = sb.set(i1, v1);
5663
5664     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
5665         sb..csize)" */
5666 }
5667
5668 static void set_set_post_s_182(ArrayList sa, ArrayList sb, int i1, Object v1, int
5669     i2, Object v2)
5670 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5671     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 =
5678         sb..csize" */
5679 }
5680
5681 static void set_set_post_c_182(ArrayList sa, ArrayList sb, int i1, Object v1, int
5682     i2, Object v2)
5683 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5684     sa..contents = sb..contents & sa..csize = sb..csize"
5685     modifies "sa..contents", "sb..contents"
5686     ensures "True" */
5687 {
5688     /*: assume "0 <= i1 & i1 < sa..csize" */
5689     Object r1a = sa.set(i1, v1);
5690     /*: assume "0 <= i2 & i2 < sa..csize" */
5691     Object r2a = sa.set(i2, v2);
5692     /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | i1 > i2)" */
5693
5694     /*: assume "0 <= i2 & i2 < sb..csize" */
5695     Object r2b = sb.set(i2, v2);
5696     /*: assume "0 <= i1 & i1 < sb..csize" */
5697     Object r1b = sb.set(i1, v1);
5698
5699     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
5700         sb..csize)" */
5701 }
5702
5703 static void set_set_pre_s_183(ArrayList sa, ArrayList sb, int i1, Object v1, int
5704     i2, Object v2)
5705 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5706     sa..contents = sb..contents & sa..csize = sb..csize"
5707     modifies "sa..contents", "sb..contents"
5708     ensures "True" */
5709 {
5710     /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
5711         sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | i1 > i2" */
5712     /*: assume "0 <= i1 & i1 < sa..csize" */
5713     Object r1a = sa.set(i1, v1);
5714     /*: assume "0 <= i2 & i2 < sa..csize" */
5715     sa.set(i2, v2);
5716
5717     sb.set(i2, v2);
5718     Object r1b = sb.set(i1, v1);
5719
5720     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
5721 }
5722
5723 static void set_set_pre_c_183(ArrayList sa, ArrayList sb, int i1, Object v1, int
5724     i2, Object v2)
5725 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5726     sa..contents = sb..contents & sa..csize = sb..csize"
5727     modifies "sa..contents", "sb..contents"
5728     ensures "True" */
5729 {

```

```

5724     /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
5725         sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | i1 > i2)" */
5726     /*: assume "0 <= i1 & i1 < sa..csize" */
5727     Object r1a = sa.set(i1, v1);
5728     /*: assume "0 <= i2 & i2 < sa..csize" */
5729     sa.set(i2, v2);
5730
5731     /*: assume "0 <= i2 & i2 < sb..csize" */
5732     sb.set(i2, v2);
5733     /*: assume "0 <= i1 & i1 < sb..csize" */
5734     Object r1b = sb.set(i1, v1);
5735
5736     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
5737         */
5738 }
5739
5740 static void set_set_between_s_184(ArrayList sa, ArrayList sb, int i1, Object v1,
5741     int i2, Object v2)
5742 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5743     sa..contents = sb..contents & sa..csize = sb..csize"
5744     modifies "sa..contents", "sb..contents"
5745     ensures "True" */
5746 {
5747     /*: assume "0 <= i1 & i1 < sa..csize" */
5748     Object r1a = sa.set(i1, v1);
5749     /*: assume "i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | i1 > i2" */
5750     /*: assume "0 <= i2 & i2 < sa..csize" */
5751     sa.set(i2, v2);
5752
5753     sb.set(i2, v2);
5754     Object r1b = sb.set(i1, v1);
5755
5756     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
5757 }
5758
5759 static void set_set_between_c_184(ArrayList sa, ArrayList sb, int i1, Object v1,
5760     int i2, Object v2)
5761 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5762     sa..contents = sb..contents & sa..csize = sb..csize"
5763     modifies "sa..contents", "sb..contents"
5764     ensures "True" */
5765 {
5766     /*: assume "0 <= i1 & i1 < sa..csize" */
5767     Object r1a = sa.set(i1, v1);
5768     /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | i1 > i2)" */
5769     /*: assume "0 <= i2 & i2 < sa..csize" */
5770     sa.set(i2, v2);
5771
5772     /*: assume "0 <= i2 & i2 < sb..csize" */
5773     sb.set(i2, v2);
5774     /*: assume "0 <= i1 & i1 < sb..csize" */
5775     Object r1b = sb.set(i1, v1);
5776
5777     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
5778         */
5779 }
5780
5781 static void set_set_post_s_185(ArrayList sa, ArrayList sb, int i1, Object v1, int
5782     i2, Object v2)
5783 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5784     sa..contents = sb..contents & sa..csize = sb..csize"
5785     modifies "sa..contents", "sb..contents"
5786     ensures "True" */
5787 {
5788     /*: 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
5795     i2, Object v2)
5796 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5797     sa..contents = sb..contents & sa..csize = sb..csize"
5798     modifies "sa..contents", "sb..contents"
5799     ensures "True" */
5800 {
5801     /*: assume "0 <= i1 & i1 < sa..csize" */
5802     Object r1a = sa.set(i1, v1);
5803     /*: assume "0 <= i2 & i2 < sa..csize" */
5804     sa.set(i2, v2);
5805     /*: assume "~(i1 < i2 | (i1 = i2 & r1a = v1 & r1a = v2 & v1 = v2) | i1 > i2)" */
5806
5807     /*: assume "0 <= i2 & i2 < sb..csize" */
5808     sb.set(i2, v2);
5809     /*: assume "0 <= i1 & i1 < sb..csize" */
5810     Object r1b = sb.set(i1, v1);
5811
5812     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
5813         */
5814 }
5815
5816 static void set_size_pre_s_186(ArrayList sa, ArrayList sb, int i1, Object v1)
5817 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5818     sa..contents = sb..contents & sa..csize = sb..csize"
5819     modifies "sa..contents", "sb..contents"
5820     ensures "True" */
5821 {
5822     /*: assume "True" */
5823     /*: assume "0 <= i1 & i1 < sa..csize" */
5824     Object r1a = sa.set(i1, v1);
5825     int r2a = sa.size();
5826
5827     int r2b = sb.size();
5828     Object r1b = sb.set(i1, v1);
5829
5830     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
5831         sb..csize" */
5832 }
5833
5834 static void set_size_pre_c_186(ArrayList sa, ArrayList sb, int i1, Object v1)
5835 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5836     sa..contents = sb..contents & sa..csize = sb..csize"
5837     modifies "sa..contents", "sb..contents"
5838     ensures "True" */
5839 {
5840     /*: assume "~(True)" */
5841     /*: assume "0 <= i1 & i1 < sa..csize" */
5842     Object r1a = sa.set(i1, v1);
5843     int r2a = sa.size();
5844
5845     int r2b = sb.size();
5846     /*: assume "0 <= i1 & i1 < sb..csize" */
5847     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 =
5917         sb..csize)" */
5918 }
5919
5920 static void set_add_at_pre_s_189(ArrayList sa, ArrayList sb, int i1, Object v1, int
5921     i2, Object v2)
5922 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5923     sa..contents = sb..contents & sa..csize = sb..csize"
5924     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
5925     "sb..msize"
5926     ensures "True" */
5927 {
5928     /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
5929         sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | (i1 > i2 & (ALL v. ((i1
5930         - 1, v) : sa..contents) = ((i1, v) : sa..contents)) & (i1 - 1, v1) :
5931         sa..contents & (i1, v1) : sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize &
5932         0 <= i1 & i1 < sa..csize)" */
5933     /*: assume "0 <= i1 & i1 < sa..csize" */
5934     sa.set(i1, v1);
5935     /*: assume "0 <= i2 & i2 <= sa..csize" */
5936     sa.add_at(i2, v2);
5937
5938     sb.add_at(i2, v2);
5939     sb.set(i1, v1);
5940
5941     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
5942 }
5943
5944 static void set_add_at_between_s_190(ArrayList sa, ArrayList sb, int i1, Object v1,
5945     int i2, Object v2)
5946 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5947     sa..contents = sb..contents & sa..csize = sb..csize"
5948     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
5949     "sb..msize"
5950     ensures "True" */
5951 {
5952     /*: assume "0 <= i1 & i1 < sa..csize" */
5953     sa.set(i1, v1);
5954     /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & (i1, v2) :
5955         sa..(old contents) & v1 = v2 & 0 <= i1 & i1 < sa..(old csize)) | (i1 > i2 &
5956         (ALL v. ((i1 - 1, v) : sa..contents) = ((i1, v) : sa..(old contents))) & (i1
5957         - 1, v1) : sa..contents & (i1, v1) : sa..(old contents) & 0 <= i1 - 1 & i1 -
5958         1 < sa..csize & 0 <= i1 & i1 < sa..(old csize))" */
5959     /*: assume "0 <= i2 & i2 <= sa..csize" */
5960     sa.add_at(i2, v2);
5961
5962     sb.add_at(i2, v2);
5963     sb.set(i1, v1);
5964
5965     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
5966 }
5967
5968 static void set_add_at_post_s_191(ArrayList sa, ArrayList sb, int i1, Object v1,
5969     int i2, Object v2)
5970 /*: 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 <
6020         sa..(old csize)) | i1 > i2" */
6021     /*: assume "0 <= i2 & i2 < sa..csize" */
6022     Object r2a = sa.get(i2);
6023
6024     Object r2b = sb.get(i2);
6025     sb.set(i1, v1);
6026
6027     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
6028 }
6029
6030 static void set_get_between_c_193(ArrayList sa, ArrayList sb, int i1, Object v1,
6031     int i2)
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 "0 <= i1 & i1 < sa..csize" */
6038     sa.set(i1, v1);
6039     /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & 0 <= i1 & i1
6040         < sa..(old csize)) | i1 > i2)" */
6041     /*: assume "0 <= i2 & i2 < sa..csize" */
6042     Object r2a = sa.get(i2);
6043
6044     /*: assume "0 <= i2 & i2 < sb..csize" */
6045     Object r2b = sb.get(i2);
6046     /*: assume "0 <= i1 & i1 < sb..csize" */
6047     sb.set(i1, v1);
6048
6049     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
6050         */
6051 }
6052
6053 static void set_get_post_s_194(ArrayList sa, ArrayList sb, int i1, Object v1, int
6054     i2)
6055 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6056     sa..contents = sb..contents & sa..csize = sb..csize"
6057     modifies "sa..contents", "sb..contents"
6058     ensures "True" */
6059 {
6060     /*: assume "0 <= i1 & i1 < sa..csize" */
6061     sa.set(i1, v1);
6062     /*: assume "0 <= i2 & i2 < sa..csize" */
6063     Object r2a = sa.get(i2);
6064     /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & 0 <= i1 & i1 <
6065         sa..(old csize)) | i1 > i2" */
6066
6067     Object r2b = sb.get(i2);
6068     sb.set(i1, v1);
6069
6070     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
6071 }
6072
6073 static void set_get_post_c_194(ArrayList sa, ArrayList sb, int i1, Object v1, int
6074     i2)
6075 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6076     sa..contents = sb..contents & sa..csize = sb..csize"
6077     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))" */
6167
        int r2b = sb.indexOf(v2);
6168
        sb.set(i1, v1);
6169
6170
        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
6171
    }
6172
6173
6174 static void set_indexOf_post_c_197(ArrayList sa, ArrayList sb, int i1, Object v1,
        Object v2)
6175
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
6176
        modifies "sa..contents", "sb..contents"
6177
        ensures "True" */
6178
    {
6179
        /*: assume "0 <= i1 & i1 < sa..csize" */
6180
        sa.set(i1, v1);
6181
        int r2a = sa.indexOf(v2);
6182
        /*: 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)))" */
6183
6184
        int r2b = sb.indexOf(v2);
6185
        /*: assume "0 <= i1 & i1 < sb..csize" */
6186
        sb.set(i1, v1);
6187
6188
        /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
6189
    }
6190
6191
6192 static void set_lastIndexOf_pre_s_198(ArrayList sa, ArrayList sb, int i1, Object
        v1, Object v2)
6193
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
6194
        modifies "sa..contents", "sb..contents"
6195
        ensures "True" */
6196
    {
6197
        /*: 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)" */
6198
6199
        /*: assume "0 <= i1 & i1 < sa..csize" */
6200
        sa.set(i1, v1);
6201
        int r2a = sa.lastIndexOf(v2);
6202
6203
        int r2b = sb.lastIndexOf(v2);
6204
        sb.set(i1, v1);
6205
6206
        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
6207
    }
6208
6209
6210 static void set_lastIndexOf_pre_c_198(ArrayList sa, ArrayList sb, int i1, Object
        v1, Object v2)
6211
    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
6212
        modifies "sa..contents", "sb..contents"
6213
        ensures "True" */
6214
    {
6215
        /*: 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))" */
6216 /*: assume "0 <= i1 & i1 < sa..csize" */
6217 sa.set(i1, v1);
6218 int r2a = sa.lastIndexOf(v2);
6219
6220 int r2b = sb.lastIndexOf(v2);
6221 /*: assume "0 <= i1 & i1 < sb..csize" */
6222 sb.set(i1, v1);
6223
6224 /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
6225 }
6226
6227 static void set_lastIndexOf_between_s_199(ArrayList sa, ArrayList sb, int i1,
        Object v1, Object v2)
6228 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6229         sa..contents = sb..contents & sa..csize = sb..csize"
6230 modifies "sa..contents", "sb..contents"
6231 ensures "True" */
6232 {
6233     /*: assume "0 <= i1 & i1 < sa..csize" */
6234     sa.set(i1, v1);
6235     /*: 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))" */
6236     int r2a = sa.lastIndexOf(v2);
6237
6238     int r2b = sb.lastIndexOf(v2);
6239     sb.set(i1, v1);
6240
6241     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
6242 }
6243
6244 static void set_lastIndexOf_between_c_199(ArrayList sa, ArrayList sb, int i1,
        Object v1, Object v2)
6245 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6246         sa..contents = sb..contents & sa..csize = sb..csize"
6247 modifies "sa..contents", "sb..contents"
6248 ensures "True" */
6249 {
6250     /*: assume "0 <= i1 & i1 < sa..csize" */
6251     sa.set(i1, v1);
6252     /*: 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))" */
6253     int r2a = sa.lastIndexOf(v2);
6254
6255     int r2b = sb.lastIndexOf(v2);
6256     /*: assume "0 <= i1 & i1 < sb..csize" */
6257     sb.set(i1, v1);
6258
6259     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */

```

```

6260 }
6261
6262 static void set_lastIndexOf_post_s_200(ArrayList sa, ArrayList sb, int i1, Object
        v1, Object v2)
6263 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6264         sa..contents = sb..contents & sa..csize = sb..csize"
6265     modifies "sa..contents", "sb..contents"
6266     ensures "True" */
6267 {
6268     /*: assume "0 <= i1 & i1 < sa..csize" */
6269     sa.set(i1, v1);
6270     int r2a = sa.lastIndexOf(v2);
6271     /*: 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" */
6272
6273     int r2b = sb.lastIndexOf(v2);
6274     sb.set(i1, v1);
6275
6276     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
6277 }
6278
6279 static void set_lastIndexOf_post_c_200(ArrayList sa, ArrayList sb, int i1, Object
        v1, Object v2)
6280 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6281         sa..contents = sb..contents & sa..csize = sb..csize"
6282     modifies "sa..contents", "sb..contents"
6283     ensures "True" */
6284 {
6285     /*: assume "0 <= i1 & i1 < sa..csize" */
6286     sa.set(i1, v1);
6287     int r2a = sa.lastIndexOf(v2);
6288     /*: 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)" */
6289
6290     int r2b = sb.lastIndexOf(v2);
6291     /*: assume "0 <= i1 & i1 < sb..csize" */
6292     sb.set(i1, v1);
6293
6294     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
6295 }
6296
6297 static void set_remove_at_pre_s_201(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2)
6298 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6299         sa..contents = sb..contents & sa..csize = sb..csize"
6300     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
6301     ensures "True" */
6302 {
6303     /*: 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)" */
6304     /*: assume "0 <= i1 & i1 < sa..csize" */
6305     sa.set(i1, v1);
6306     /*: assume "0 <= i2 & i2 < sa..csize" */
6307     Object r2a = sa.remove_at(i2);
6308

```

```

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
6316     v1, int i2)
6317 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6318     sa..contents = sb..contents & sa..csize = sb..csize"
6319     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
6320     ensures "True" */
6321 {
6322     /*: assume "0 <= i1 & i1 < sa..csize" */
6323     sa.set(i1, v1);
6324     /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (ALL v. ((i1, v) :
6325     sa..(old contents)) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..(old
6326     contents) & (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) & 0
6327     <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v.
6328     ((i1, v) : sa..(old contents)) = ((i1 + 1, v) : sa..contents)) & (i1, v1) :
6329     sa..(old contents) & (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old
6330     csize) & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
6331     /*: assume "0 <= i2 & i2 < sa..csize" */
6332     Object r2a = sa.remove_at(i2);
6333
6334     Object r2b = sb.remove_at(i2);
6335     sb.set(i1, v1);
6336
6337     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
6338 }
6339
6340 static void set_remove_at_post_s_203(ArrayList sa, ArrayList sb, int i1, Object v1,
6341     int i2)
6342 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6343     sa..contents = sb..contents & sa..csize = sb..csize"
6344     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
6345     ensures "True" */
6346 {
6347     /*: assume "0 <= i1 & i1 < sa..csize" */
6348     sa.set(i1, v1);
6349     /*: assume "0 <= i2 & i2 < sa..csize" */
6350     Object r2a = sa.remove_at(i2);
6351     /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize & (ALL v. ((i1, v) : sa..(old
6352     contents)) = ((i1, v) : sa..contents)) & (i1, v1) : sa..(old contents) &
6353     (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 & i1 <
6354     sa..csize) | (sa..csize > i1 & i1 > i2 & (ALL v. ((i1, v) : sa..(old
6355     contents)) = ((i1, v) : sa..contents)) & (i1, v1) : sa..(old contents) &
6356     (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 & i1 <
6357     sa..csize)" */
6358
6359     Object r2b = sb.remove_at(i2);
6360     sb.set(i1, v1);
6361
6362     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
6363 }
6364
6365 static void set_remove_at_pre_s_204(ArrayList sa, ArrayList sb, int i1, Object v1,
6366     int i2)
6367 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6368     sa..contents = sb..contents & sa..csize = sb..csize"
6369     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
6370     ensures "True" */
6371 {
6372     /*: assume "i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, v1) :
6373     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 &
        sa..contents = sb..contents & sa..csize = sb..csize"
6425     modifies "sa..contents", "sb..contents"
6426     ensures "True" */
6427 {
6428     /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
        sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | i1 > i2)" */
6429     /*: assume "0 <= i1 & i1 < sa..csize" */
6430     sa.set(i1, v1);
6431     /*: assume "0 <= i2 & i2 < sa..csize" */
6432     Object r2a = sa.set(i2, v2);
6433
6434     /*: assume "0 <= i2 & i2 < sb..csize" */
6435     Object r2b = sb.set(i2, v2);
6436     /*: assume "0 <= i1 & i1 < sb..csize" */
6437     sb.set(i1, v1);
6438
6439     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
6440 }
6441
6442 static void set_set_between_s_208(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2, Object v2)
6443 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
6444     modifies "sa..contents", "sb..contents"
6445     ensures "True" */
6446 {
6447     /*: assume "0 <= i1 & i1 < sa..csize" */
6448     sa.set(i1, v1);
6449     /*: 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" */
6450     /*: assume "0 <= i2 & i2 < sa..csize" */
6451     Object r2a = sa.set(i2, v2);
6452
6453     Object r2b = sb.set(i2, v2);
6454     sb.set(i1, v1);
6455
6456     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
6457 }
6458
6459 static void set_set_between_c_208(ArrayList sa, ArrayList sb, int i1, Object v1,
        int i2, Object v2)
6460 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
6461     modifies "sa..contents", "sb..contents"
6462     ensures "True" */
6463 {

```

```

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) :
6470     sa..(old contents) & v1 = v2 & 0 <= i1 & i1 < sa..(old csize)) | i1 > i2)" */
6471     /*: assume "0 <= i2 & i2 < sa..csize" */
6472     Object r2a = sa.set(i2, v2);
6473
6474     /*: assume "0 <= i2 & i2 < sb..csize" */
6475     Object r2b = sb.set(i2, v2);
6476     /*: assume "0 <= i1 & i1 < sb..csize" */
6477     sb.set(i1, v1);
6478     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
6479     */
6480 }
6481
6482 static void set_set_post_s_209(ArrayList sa, ArrayList sb, int i1, Object v1, int
6483     i2, Object v2)
6484 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6485     sa..contents = sb..contents & sa..csize = sb..csize"
6486     modifies "sa..contents", "sb..contents"
6487     ensures "True" */
6488 {
6489     /*: assume "0 <= i1 & i1 < sa..csize" */
6490     sa.set(i1, v1);
6491     /*: assume "0 <= i2 & i2 < sa..csize" */
6492     Object r2a = sa.set(i2, v2);
6493     /*: assume "i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & (i1, v2) :
6494     sa..(old contents) & v1 = v2 & 0 <= i1 & i1 < sa..(old csize)) | i1 > i2" */
6495     Object r2b = sb.set(i2, v2);
6496     sb.set(i1, v1);
6497
6498     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
6499 }
6500
6501 static void set_set_post_c_209(ArrayList sa, ArrayList sb, int i1, Object v1, int
6502     i2, Object v2)
6503 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6504     sa..contents = sb..contents & sa..csize = sb..csize"
6505     modifies "sa..contents", "sb..contents"
6506     ensures "True" */
6507 {
6508     /*: assume "0 <= i1 & i1 < sa..csize" */
6509     sa.set(i1, v1);
6510     /*: assume "0 <= i2 & i2 < sa..csize" */
6511     Object r2a = sa.set(i2, v2);
6512     /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & (i1, v2) :
6513     sa..(old contents) & v1 = v2 & 0 <= i1 & i1 < sa..(old csize)) | i1 > i2)" */
6514     /*: assume "0 <= i2 & i2 < sb..csize" */
6515     Object r2b = sb.set(i2, v2);
6516     /*: assume "0 <= i1 & i1 < sb..csize" */
6517     sb.set(i1, v1);
6518
6519     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
6520     */
6521 }
6522
6523 static void set_set_pre_s_210(ArrayList sa, ArrayList sb, int i1, Object v1, int
6524     i2, Object v2)
6525 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6526     sa..contents = sb..contents & sa..csize = sb..csize"
6527     modifies "sa..contents", "sb..contents"
6528     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 &
        sa..contents = sb..contents & sa..csize = sb..csize"
6539 modifies "sa..contents", "sb..contents"
6540 ensures "True" */
6541 {
6542     /*: assume "~(i1 < i2 | (i1 = i2 & v1 = v2) | i1 > i2)" */
6543     /*: assume "0 <= i1 & i1 < sa..csize" */
6544     sa.set(i1, v1);
6545     /*: assume "0 <= i2 & i2 < sa..csize" */
6546     sa.set(i2, v2);
6547
6548     /*: assume "0 <= i2 & i2 < sb..csize" */
6549     sb.set(i2, v2);
6550     /*: assume "0 <= i1 & i1 < sb..csize" */
6551     sb.set(i1, v1);
6552
6553     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
6554 }
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 &
        sa..contents = sb..contents & sa..csize = sb..csize"
6559 modifies "sa..contents", "sb..contents"
6560 ensures "True" */
6561 {
6562     /*: assume "0 <= i1 & i1 < sa..csize" */
6563     sa.set(i1, v1);
6564     /*: assume "i1 < i2 | (i1 = i2 & v1 = v2) | i1 > i2" */
6565     /*: assume "0 <= i2 & i2 < sa..csize" */
6566     sa.set(i2, v2);
6567
6568     sb.set(i2, v2);
6569     sb.set(i1, v1);
6570
6571     /*: assert "sa..contents = sb..contents & sa..csize = sb..csize" */
6572 }
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 &
        sa..contents = sb..contents & sa..csize = sb..csize"
6577 modifies "sa..contents", "sb..contents"
6578 ensures "True" */
6579 {
6580     /*: assume "0 <= i1 & i1 < sa..csize" */
6581     sa.set(i1, v1);
6582     /*: assume "~(i1 < i2 | (i1 = i2 & v1 = v2) | i1 > i2)" */
6583     /*: assume "0 <= i2 & i2 < sa..csize" */
6584     sa.set(i2, v2);
6585

```

```

6586     /*: assume "0 <= i2 & i2 < sb..csize" */
6587     sb.set(i2, v2);
6588     /*: assume "0 <= i1 & i1 < sb..csize" */
6589     sb.set(i1, v1);
6590
6591     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
6592 }
6593
6594 static void set_set_post_s_212(ArrayList sa, ArrayList sb, int i1, Object v1, int
6595     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
6614     i2, Object v2)
6615 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6616     sa..contents = sb..contents & sa..csize = sb..csize"
6617     modifies "sa..contents", "sb..contents"
6618     ensures "True" */
6619 {
6620     /*: assume "0 <= i1 & i1 < sa..csize" */
6621     sa.set(i1, v1);
6622     /*: assume "0 <= i2 & i2 < sa..csize" */
6623     sa.set(i2, v2);
6624     /*: assume "~(i1 < i2 | (i1 = i2 & v1 = v2) | i1 > i2)" */
6625
6626     /*: assume "0 <= i2 & i2 < sb..csize" */
6627     sb.set(i2, v2);
6628     /*: assume "0 <= i1 & i1 < sb..csize" */
6629     sb.set(i1, v1);
6630
6631     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
6632 }
6633
6634 static void set_size_pre_s_213(ArrayList sa, ArrayList sb, int i1, Object v1)
6635 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6636     sa..contents = sb..contents & sa..csize = sb..csize"
6637     modifies "sa..contents", "sb..contents"
6638     ensures "True" */
6639 {
6640     /*: assume "True" */
6641     /*: assume "0 <= i1 & i1 < sa..csize" */
6642     sa.set(i1, v1);
6643     int r2a = sa.size();
6644
6645     int r2b = sb.size();
6646     sb.set(i1, v1);
6647
6648     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
6649 }

```

```

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
6669 static void set_size_between_s_214(ArrayList sa, ArrayList sb, int i1, Object v1)
6670 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6671      sa..contents = sb..contents & sa..csize = sb..csize"
6672      modifies "sa..contents", "sb..contents"
6673      ensures "True" */
6674 {
6675     /*: assume "0 <= i1 & i1 < sa..csize" */
6676     sa.set(i1, v1);
6677     /*: assume "True" */
6678     int r2a = sa.size();
6679
6680     int r2b = sb.size();
6681     sb.set(i1, v1);
6682
6683     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
6684 }
6685
6686 static void set_size_between_c_214(ArrayList sa, ArrayList sb, int i1, Object v1)
6687 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6688      sa..contents = sb..contents & sa..csize = sb..csize"
6689      modifies "sa..contents", "sb..contents"
6690      ensures "True" */
6691 {
6692     /*: assume "0 <= i1 & i1 < sa..csize" */
6693     sa.set(i1, v1);
6694     /*: assume "~(True)" */
6695     int r2a = sa.size();
6696
6697     int r2b = sb.size();
6698     /*: assume "0 <= i1 & i1 < sb..csize" */
6699     sb.set(i1, v1);
6700
6701     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
6702         */
6703 }
6704
6705 static void set_size_post_s_215(ArrayList sa, ArrayList sb, int i1, Object v1)
6706 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6707      sa..contents = sb..contents & sa..csize = sb..csize"
6708      modifies "sa..contents", "sb..contents"
6709      ensures "True" */
6710 {
6711     /*: assume "0 <= i1 & i1 < sa..csize" */
6712     sa.set(i1, v1);
6713     int r2a = sa.size();

```

```

6712     /*: assume "True" */
6713
6714     int r2b = sb.size();
6715     sb.set(i1, v1);
6716
6717     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
6718 }
6719
6720 static void set_size_post_c_215(ArrayList sa, ArrayList sb, int i1, Object v1)
6721 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6722     sa..contents = sb..contents & sa..csize = sb..csize"
6723     modifies "sa..contents", "sb..contents"
6724     ensures "True" */
6725 {
6726     /*: assume "0 <= i1 & i1 < sa..csize" */
6727     sa.set(i1, v1);
6728     int r2a = sa.size();
6729     /*: assume "~(True)" */
6730
6731     int r2b = sb.size();
6732     /*: assume "0 <= i1 & i1 < sb..csize" */
6733     sb.set(i1, v1);
6734
6735     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
6736         */
6737 }
6738
6739 static void size_add_at_pre_s_216(ArrayList sa, ArrayList sb, int i2, Object v2)
6740 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6741     sa..contents = sb..contents & sa..csize = sb..csize"
6742     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
6743     "sb..msize"
6744     ensures "True" */
6745 {
6746     /*: assume "False" */
6747     int r1a = sa.size();
6748     /*: assume "0 <= i2 & i2 <= sa..csize" */
6749     sa.add_at(i2, v2);
6750
6751     sb.add_at(i2, v2);
6752     int r1b = sb.size();
6753
6754     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
6755 }
6756
6757 static void size_add_at_pre_c_216(ArrayList sa, ArrayList sb, int i2, Object v2)
6758 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6759     sa..contents = sb..contents & sa..csize = sb..csize"
6760     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
6761     "sb..msize"
6762     ensures "True" */
6763 {
6764     /*: assume "~(False)" */
6765     int r1a = sa.size();
6766     /*: assume "0 <= i2 & i2 <= sa..csize" */
6767     sa.add_at(i2, v2);
6768
6769     /*: assume "0 <= i2 & i2 <= sb..csize" */
6770     sb.add_at(i2, v2);
6771     int r1b = sb.size();
6772
6773     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
6774         */
6775 }

```

```

6773 static void size_add_at_between_s_217(ArrayList sa, ArrayList sb, int i2, Object v2)
6774 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6775     sa..contents = sb..contents & sa..csize = sb..csize"
6776     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
        "sb..msize"
6777     ensures "True" */
6778 {
6779     int r1a = sa.size();
6780     /*: assume "False" */
6781     /*: assume "0 <= i2 & i2 <= sa..csize" */
6782     sa.add_at(i2, v2);
6783
6784     sb.add_at(i2, v2);
6785     int r1b = sb.size();
6786
6787     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
6788 }
6789
6790 static void size_add_at_between_c_217(ArrayList sa, ArrayList sb, int i2, Object v2)
6791 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6792     sa..contents = sb..contents & sa..csize = sb..csize"
6793     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
        "sb..msize"
6794     ensures "True" */
6795 {
6796     int r1a = sa.size();
6797     /*: assume "~(False)" */
6798     /*: assume "0 <= i2 & i2 <= sa..csize" */
6799     sa.add_at(i2, v2);
6800
6801     /*: assume "0 <= i2 & i2 <= sb..csize" */
6802     sb.add_at(i2, v2);
6803     int r1b = sb.size();
6804
6805     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
6806 }
6807
6808 static void size_add_at_post_s_218(ArrayList sa, ArrayList sb, int i2, Object v2)
6809 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6810     sa..contents = sb..contents & sa..csize = sb..csize"
6811     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
        "sb..msize"
6812     ensures "True" */
6813 {
6814     int r1a = sa.size();
6815     /*: assume "0 <= i2 & i2 <= sa..csize" */
6816     sa.add_at(i2, v2);
6817     /*: assume "False" */
6818
6819     sb.add_at(i2, v2);
6820     int r1b = sb.size();
6821
6822     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
6823 }
6824
6825 static void size_add_at_post_c_218(ArrayList sa, ArrayList sb, int i2, Object v2)
6826 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6827     sa..contents = sb..contents & sa..csize = sb..csize"
6828     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
        "sb..msize"
6829     ensures "True" */
6830 {
6831     int r1a = sa.size();
6832     /*: 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 &
        sa..contents = sb..contents & sa..csize = sb..csize"
6911     ensures "True" */
6912 {
6913     int r1a = sa.size();
6914     /*: assume "0 <= i2 & i2 < sa..csize" */
6915     Object r2a = sa.get(i2);
6916     /*: assume "True" */
6917
6918     Object r2b = sb.get(i2);
6919     int r1b = sb.size();
6920
6921     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
6922 }
6923
6924 static void size_get_post_c_221(ArrayList sa, ArrayList sb, int i2)
6925 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
6926     ensures "True" */
6927 {
6928     int r1a = sa.size();
6929     /*: assume "0 <= i2 & i2 < sa..csize" */
6930     Object r2a = sa.get(i2);
6931     /*: assume "~(True)" */
6932
6933     /*: assume "0 <= i2 & i2 < sb..csize" */
6934     Object r2b = sb.get(i2);
6935     int r1b = sb.size();
6936
6937     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
6938 }
6939
6940 static void size_index0f_pre_s_222(ArrayList sa, ArrayList sb, Object v2)
6941 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
        sa..contents = sb..contents & sa..csize = sb..csize"
6942     ensures "True" */
6943 {
6944     /*: assume "True" */
6945     int r1a = sa.size();
6946     int r2a = sa.index0f(v2);
6947
6948     int r2b = sb.index0f(v2);
6949     int r1b = sb.size();
6950
6951     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
6952 }

```

```

6955 }
6956
6957 static void size_indexOf_pre_c_222(ArrayList sa, ArrayList sb, Object v2)
6958 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6959           sa..contents = sb..contents & sa..csize = sb..csize"
6960     ensures "True" */
6961 {
6962     /*: assume "~(True)" */
6963     int r1a = sa.size();
6964     int r2a = sa.indexOf(v2);
6965
6966     int r2b = sb.indexOf(v2);
6967     int r1b = sb.size();
6968
6969     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
6970 }
6971
6972 static void size_indexOf_between_s_223(ArrayList sa, ArrayList sb, Object v2)
6973 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6974           sa..contents = sb..contents & sa..csize = sb..csize"
6975     ensures "True" */
6976 {
6977     int r1a = sa.size();
6978     /*: assume "True" */
6979     int r2a = sa.indexOf(v2);
6980
6981     int r2b = sb.indexOf(v2);
6982     int r1b = sb.size();
6983
6984     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
6985 }
6986
6987 static void size_indexOf_between_c_223(ArrayList sa, ArrayList sb, Object v2)
6988 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
6989           sa..contents = sb..contents & sa..csize = sb..csize"
6990     ensures "True" */
6991 {
6992     int r1a = sa.size();
6993     /*: assume "~(True)" */
6994     int r2a = sa.indexOf(v2);
6995
6996     int r2b = sb.indexOf(v2);
6997     int r1b = sb.size();
6998
6999     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize)" */
7000 }
7001
7002 static void size_indexOf_post_s_224(ArrayList sa, ArrayList sb, Object v2)
7003 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7004           sa..contents = sb..contents & sa..csize = sb..csize"
7005     ensures "True" */
7006 {
7007     int r1a = sa.size();
7008     int r2a = sa.indexOf(v2);
7009     /*: assume "True" */
7010
7011     int r2b = sb.indexOf(v2);
7012     int r1b = sb.size();
7013
7014     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
        sb..csize" */
7015 }

```

```

7016 static void size_indexOf_post_c_224(ArrayList sa, ArrayList sb, Object v2)
7017 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7018 sa..contents = sb..contents & sa..csize = sb..csize"
7019 ensures "True" */
7020 {
7021     int r1a = sa.size();
7022     int r2a = sa.indexOf(v2);
7023     /*: assume "~(True)" */
7024
7025     int r2b = sb.indexOf(v2);
7026     int r1b = sb.size();
7027
7028     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
7029 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 =
7045 sb..csize" */
7046 }
7047
7048 static void size_lastIndexOf_pre_c_225(ArrayList sa, ArrayList sb, Object v2)
7049 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7050 sa..contents = sb..contents & sa..csize = sb..csize"
7051 ensures "True" */
7052 {
7053     /*: assume "~(True)" */
7054     int r1a = sa.size();
7055     int r2a = sa.lastIndexOf(v2);
7056
7057     int r2b = sb.lastIndexOf(v2);
7058     int r1b = sb.size();
7059
7060     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
7061 sb..csize)" */
7062 }
7063
7064 static void size_lastIndexOf_between_s_226(ArrayList sa, ArrayList sb, Object v2)
7065 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7066 sa..contents = sb..contents & sa..csize = sb..csize"
7067 ensures "True" */
7068 {
7069     int r1a = sa.size();
7070     /*: assume "True" */
7071     int r2a = sa.lastIndexOf(v2);
7072
7073     int r2b = sb.lastIndexOf(v2);
7074     int r1b = sb.size();
7075
7076     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
7077 sb..csize" */
7078 }

```

```

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 =
7090          sb..csize)" */
7091 }
7092
7093 static void size_lastIndexOf_post_s_227(ArrayList sa, ArrayList sb, Object v2)
7094 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7095          sa..contents = sb..contents & sa..csize = sb..csize"
7096 ensures "True" */
7097 {
7098     int r1a = sa.size();
7099     int r2a = sa.lastIndexOf(v2);
7100     /*: assume "True" */
7101
7102     int r2b = sb.lastIndexOf(v2);
7103     int r1b = sb.size();
7104
7105     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
7106          sb..csize" */
7107 }
7108
7109 static void size_lastIndexOf_post_c_227(ArrayList sa, ArrayList sb, Object v2)
7110 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7111          sa..contents = sb..contents & sa..csize = sb..csize"
7112 ensures "True" */
7113 {
7114     int r1a = sa.size();
7115     int r2a = sa.lastIndexOf(v2);
7116     /*: assume "~(True)" */
7117
7118     int r2b = sb.lastIndexOf(v2);
7119     int r1b = sb.size();
7120
7121     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
7122          sb..csize)" */
7123 }
7124
7125 static void size_remove_at_pre_s_228(ArrayList sa, ArrayList sb, int i2)
7126 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7127          sa..contents = sb..contents & sa..csize = sb..csize"
7128 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7129 ensures "True" */
7130 {
7131     /*: assume "False" */
7132     int r1a = sa.size();
7133     /*: assume "0 <= i2 & i2 < sa..csize" */
7134     Object r2a = sa.remove_at(i2);
7135
7136     Object r2b = sb.remove_at(i2);
7137     int r1b = sb.size();
7138
7139     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
7140          sb..csize" */
7141 }

```

```

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 =
7155      sb..csize)" */
7156 }
7157
7158 static void size_remove_at_between_s_229(ArrayList sa, ArrayList sb, int i2)
7159 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7160      sa..contents = sb..contents & sa..csize = sb..csize"
7161      modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7162      ensures "True" */
7163 {
7164     int r1a = sa.size();
7165     /*: assume "False" */
7166     /*: assume "0 <= i2 & i2 < sa..csize" */
7167     Object r2a = sa.remove_at(i2);
7168
7169     Object r2b = sb.remove_at(i2);
7170     int r1b = sb.size();
7171
7172     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
7173      sb..csize" */
7174 }
7175
7176 static void size_remove_at_between_c_229(ArrayList sa, ArrayList sb, int i2)
7177 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7178      sa..contents = sb..contents & sa..csize = sb..csize"
7179      modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7180      ensures "True" */
7181 {
7182     int r1a = sa.size();
7183     /*: assume "~(False)" */
7184     /*: assume "0 <= i2 & i2 < sa..csize" */
7185     Object r2a = sa.remove_at(i2);
7186
7187     /*: assume "0 <= i2 & i2 < sb..csize" */
7188     Object r2b = sb.remove_at(i2);
7189     int r1b = sb.size();
7190
7191     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
7192      sb..csize)" */
7193 }
7194
7195 static void size_remove_at_post_s_230(ArrayList sa, ArrayList sb, int i2)
7196 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7197      sa..contents = sb..contents & sa..csize = sb..csize"
7198      modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7199      ensures "True" */
7200 {
7201     int r1a = sa.size();
7202     /*: 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

```

```

7262 static void size_remove_at_between_s_232(ArrayList sa, ArrayList sb, int i2)
7263 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7264      sa..contents = sb..contents & sa..csize = sb..csize"
7265      modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7266      ensures "True" */
7267 {
7268     int r1a = sa.size();
7269     /*: assume "False" */
7270     /*: assume "0 <= i2 & i2 < sa..csize" */
7271     sa.remove_at(i2);
7272
7273     sb.remove_at(i2);
7274     int r1b = sb.size();
7275
7276     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
7277 }
7278
7279 static void size_remove_at_between_c_232(ArrayList sa, ArrayList sb, 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     int r1a = sa.size();
7286     /*: assume "~(False)" */
7287     /*: assume "0 <= i2 & i2 < sa..csize" */
7288     sa.remove_at(i2);
7289
7290     /*: assume "0 <= i2 & i2 < sb..csize" */
7291     sb.remove_at(i2);
7292     int r1b = sb.size();
7293
7294     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
7295      */
7296 }
7297
7298 static void size_remove_at_post_s_233(ArrayList sa, ArrayList sb, int i2)
7299 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7300      sa..contents = sb..contents & sa..csize = sb..csize"
7301      modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7302      ensures "True" */
7303 {
7304     int r1a = sa.size();
7305     /*: assume "0 <= i2 & i2 < sa..csize" */
7306     sa.remove_at(i2);
7307     /*: assume "False" */
7308
7309     sb.remove_at(i2);
7310     int r1b = sb.size();
7311
7312     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
7313 }
7314
7315 static void size_remove_at_post_c_233(ArrayList sa, ArrayList sb, int i2)
7316 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7317      sa..contents = sb..contents & sa..csize = sb..csize"
7318      modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
7319      ensures "True" */
7320 {
7321     int r1a = sa.size();
7322     /*: assume "0 <= i2 & i2 < sa..csize" */
7323     sa.remove_at(i2);
7324     /*: assume "~(False)" */
7325
7326     sb.remove_at(i2);
7327     int r1b = sb.size();
7328
7329     /*: assume "0 <= i2 & i2 < sb..csize" */

```



```

7326     sb.remove_at(i2);
7327     int r1b = sb.size();
7328
7329     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
       */
7330 }
7331
7332 static void size_set_pre_s_234(ArrayList sa, ArrayList sb, int i2, Object v2)
7333 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7334             sa..contents = sb..contents & sa..csize = sb..csize"
7335   modifies "sa..contents", "sb..contents"
7336   ensures "True" */
7337 {
7338     /*: assume "True" */
7339     int r1a = sa.size();
7340     /*: assume "0 <= i2 & i2 < sa..csize" */
7341     Object r2a = sa.set(i2, v2);
7342
7343     Object r2b = sb.set(i2, v2);
7344     int r1b = sb.size();
7345
7346     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
       sb..csize" */
7347 }
7348
7349 static void size_set_pre_c_234(ArrayList sa, ArrayList sb, int i2, Object v2)
7350 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7351             sa..contents = sb..contents & sa..csize = sb..csize"
7352   modifies "sa..contents", "sb..contents"
7353   ensures "True" */
7354 {
7355     /*: assume "~(True)" */
7356     int r1a = sa.size();
7357     /*: assume "0 <= i2 & i2 < sa..csize" */
7358     Object r2a = sa.set(i2, v2);
7359
7360     /*: assume "0 <= i2 & i2 < sb..csize" */
7361     Object r2b = sb.set(i2, v2);
7362     int r1b = sb.size();
7363
7364     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
       sb..csize)" */
7365 }
7366
7367 static void size_set_between_s_235(ArrayList sa, ArrayList sb, int i2, Object v2)
7368 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7369             sa..contents = sb..contents & sa..csize = sb..csize"
7370   modifies "sa..contents", "sb..contents"
7371   ensures "True" */
7372 {
7373     int r1a = sa.size();
7374     /*: assume "True" */
7375     /*: assume "0 <= i2 & i2 < sa..csize" */
7376     Object r2a = sa.set(i2, v2);
7377
7378     Object r2b = sb.set(i2, v2);
7379     int r1b = sb.size();
7380
7381     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
       sb..csize" */
7382 }
7383
7384 static void size_set_between_c_235(ArrayList sa, ArrayList sb, int i2, Object v2)
7385 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7386             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 =
7400         sb..csize)" */
7401 }
7402
7403 static void size_set_post_s_236(ArrayList sa, ArrayList sb, int i2, Object v2)
7404 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7405     sa..contents = sb..contents & sa..csize = sb..csize"
7406     modifies "sa..contents", "sb..contents"
7407     ensures "True" */
7408 {
7409     int r1a = sa.size();
7410     /*: assume "0 <= i2 & i2 < sa..csize" */
7411     Object r2a = sa.set(i2, v2);
7412     /*: assume "True" */
7413
7414     Object r2b = sb.set(i2, v2);
7415     int r1b = sb.size();
7416
7417     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
7418         sb..csize" */
7419 }
7420
7421 static void size_set_post_c_236(ArrayList sa, ArrayList sb, int i2, Object v2)
7422 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7423     sa..contents = sb..contents & sa..csize = sb..csize"
7424     modifies "sa..contents", "sb..contents"
7425     ensures "True" */
7426 {
7427     int r1a = sa.size();
7428     /*: assume "0 <= i2 & i2 < sa..csize" */
7429     Object r2a = sa.set(i2, v2);
7430     /*: assume "~(True)" */
7431
7432     /*: assume "0 <= i2 & i2 < sb..csize" */
7433     Object r2b = sb.set(i2, v2);
7434     int r1b = sb.size();
7435
7436     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
7437         sb..csize)" */
7438 }
7439
7440 static void size_set_pre_s_237(ArrayList sa, ArrayList sb, int i2, Object v2)
7441 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7442     sa..contents = sb..contents & sa..csize = sb..csize"
7443     modifies "sa..contents", "sb..contents"
7444     ensures "True" */
7445 {
7446     /*: assume "True" */
7447     int r1a = sa.size();
7448     /*: assume "0 <= i2 & i2 < sa..csize" */
7449     sa.set(i2, v2);
7450
7451     sb.set(i2, v2);

```

```

7449     int r1b = sb.size();
7450
7451     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
7452 }
7453
7454 static void size_set_pre_c_237(ArrayList sa, ArrayList sb, int i2, Object v2)
7455 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7456     sa..contents = sb..contents & sa..csize = sb..csize"
7457     modifies "sa..contents", "sb..contents"
7458     ensures "True" */
7459 {
7460     /*: assume "~(True)" */
7461     int r1a = sa.size();
7462     /*: assume "0 <= i2 & i2 < sa..csize" */
7463     sa.set(i2, v2);
7464
7465     /*: assume "0 <= i2 & i2 < sb..csize" */
7466     sb.set(i2, v2);
7467     int r1b = sb.size();
7468
7469     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
7470         */
7471 }
7472
7473 static void size_set_between_s_238(ArrayList sa, ArrayList sb, int i2, Object v2)
7474 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7475     sa..contents = sb..contents & sa..csize = sb..csize"
7476     modifies "sa..contents", "sb..contents"
7477     ensures "True" */
7478 {
7479     int r1a = sa.size();
7480     /*: assume "True" */
7481     /*: assume "0 <= i2 & i2 < sa..csize" */
7482     sa.set(i2, v2);
7483
7484     sb.set(i2, v2);
7485     int r1b = sb.size();
7486
7487     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
7488 }
7489
7490 static void size_set_between_c_238(ArrayList sa, ArrayList sb, int i2, Object v2)
7491 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7492     sa..contents = sb..contents & sa..csize = sb..csize"
7493     modifies "sa..contents", "sb..contents"
7494     ensures "True" */
7495 {
7496     int r1a = sa.size();
7497     /*: assume "~(True)" */
7498     /*: assume "0 <= i2 & i2 < sa..csize" */
7499     sa.set(i2, v2);
7500
7501     /*: assume "0 <= i2 & i2 < sb..csize" */
7502     sb.set(i2, v2);
7503     int r1b = sb.size();
7504
7505     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
7506         */
7507 }
7508
7509 static void size_set_post_s_239(ArrayList sa, ArrayList sb, int i2, Object v2)
7510 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7511     sa..contents = sb..contents & sa..csize = sb..csize"
7512     modifies "sa..contents", "sb..contents"
7513     ensures "True" */

```

```

7512 {
7513     int r1a = sa.size();
7514     /*: assume "0 <= i2 & i2 < sa..csize" */
7515     sa.set(i2, v2);
7516     /*: assume "True" */
7517
7518     sb.set(i2, v2);
7519     int r1b = sb.size();
7520
7521     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
7522 }
7523
7524 static void size_set_post_c_239(ArrayList sa, ArrayList sb, int i2, Object v2)
7525 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7526     sa..contents = sb..contents & sa..csize = sb..csize"
7527     modifies "sa..contents", "sb..contents"
7528     ensures "True" */
7529 {
7530     int r1a = sa.size();
7531     /*: assume "0 <= i2 & i2 < sa..csize" */
7532     sa.set(i2, v2);
7533     /*: assume "~(True)" */
7534
7535     /*: assume "0 <= i2 & i2 < sb..csize" */
7536     sb.set(i2, v2);
7537     int r1b = sb.size();
7538
7539     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
7540         */
7541 }
7542
7543 static void size_size_pre_s_240(ArrayList sa, ArrayList sb)
7544 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7545     sa..contents = sb..contents & sa..csize = sb..csize"
7546     ensures "True" */
7547 {
7548     /*: assume "True" */
7549     int r1a = sa.size();
7550     int r2a = sa.size();
7551
7552     int r2b = sb.size();
7553     int r1b = sb.size();
7554
7555     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
7556         sb..csize" */
7557 }
7558
7559 static void size_size_pre_c_240(ArrayList sa, ArrayList sb)
7560 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7561     sa..contents = sb..contents & sa..csize = sb..csize"
7562     ensures "True" */
7563 {
7564     /*: assume "~(True)" */
7565     int r1a = sa.size();
7566     int r2a = sa.size();
7567
7568     int r2b = sb.size();
7569     int r1b = sb.size();
7570
7571     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
7572         sb..csize)" */
7573 }
7574
7575 static void size_size_between_s_241(ArrayList sa, ArrayList sb)
7576 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &

```

```

7574         sa..contents = sb..contents & sa..csize = sb..csize"
7575     ensures "True" */
7576 {
7577     int r1a = sa.size();
7578     /*: assume "True" */
7579     int r2a = sa.size();
7580
7581     int r2b = sb.size();
7582     int r1b = sb.size();
7583
7584     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
7585         sb..csize" */
7586 }
7587
7588 static void size_size_between_c_241(ArrayList sa, ArrayList sb)
7589 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7590     sa..contents = sb..contents & sa..csize = sb..csize"
7591     ensures "True" */
7592 {
7593     int r1a = sa.size();
7594     /*: assume "~(True)" */
7595     int r2a = sa.size();
7596
7597     int r2b = sb.size();
7598     int r1b = sb.size();
7599
7600     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
7601         sb..csize)" */
7602 }
7603
7604 static void size_size_post_s_242(ArrayList sa, ArrayList sb)
7605 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7606     sa..contents = sb..contents & sa..csize = sb..csize"
7607     ensures "True" */
7608 {
7609     int r1a = sa.size();
7610     int r2a = sa.size();
7611     /*: assume "True" */
7612
7613     int r2b = sb.size();
7614     int r1b = sb.size();
7615
7616     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
7617         sb..csize" */
7618 }
7619
7620 static void size_size_post_c_242(ArrayList sa, ArrayList sb)
7621 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
7622     sa..contents = sb..contents & sa..csize = sb..csize"
7623     ensures "True" */
7624 {
7625     int r1a = sa.size();
7626     int r2a = sa.size();
7627     /*: assume "~(True)" */
7628
7629     int r2b = sb.size();
7630     int r1b = sb.size();
7631
7632     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
7633         sb..csize)" */
7634 }
7635 }

```

Listing 19. ArrayListComm.java

```

1 class ArrayListComm {
2     static void add_at_index0f_between_s_7(ArrayList sa, ArrayList sb, int i1, Object
3         v1, Object v2)
4     /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
5         sa..contents = sb..contents & sa..csize = sb..csize"
6         modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
7         "sb..msize"
8         ensures "True" */
9     {
10        /*: assume "0 <= i1 & i1 <= sa..csize" */
11        sa.add_at(i1, v1);
12        /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
13            (i, v2) : sa..contents & 0 <= i & i < i1) | ~(EX i. (i, v2) : sa..contents
14            & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize &
15            (i1 + 1, v2) : sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
16        {
17            /*: assuming "(EX i. (i, v2) : sa..(old contents) & i1 <= i & i < sa..(old
18                csize))" */
19            {
20                /*: pickWitness j :: int suchThat "(j, v2) : sa..(old contents) & i1 <=
21                    j & j < sa..(old csize)" */
22                /*: note "(j + 1, v2) : sa..contents & i1 + 1 <= j + 1 & j + 1 <
23                    sa..csize" */
24                /*: note "(EX i. (i, v2) : sa..contents & i1 + 1 <= i & i < sa..csize)"
25                    */
26            }
27            /*: note "(EX i. (i, v2) : sa..contents & i1 + 1 <= i & i < sa..csize)" */
28        }
29        int r2a = sa.index0f(v2);
30
31        int r2b = sb.index0f(v2);
32        sb.add_at(i1, v1);
33
34        /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
35    }
36
37    static void add_at_index0f_post_s_8(ArrayList sa, ArrayList sb, int i1, Object v1,
38        Object v2)
39    /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
40        sa..contents = sb..contents & sa..csize = sb..csize"
41        modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
42        "sb..msize"
43        ensures "True" */
44    {
45        /*: assume "0 <= i1 & i1 <= sa..csize" */
46        sa.add_at(i1, v1);
47        {
48            /*: assuming "(EX i. (i, v2) : sa..(old contents) & i1 <= i & i < sa..(old
49                csize))" */
50            {
51                /*: pickWitness j :: int suchThat "(j, v2) : sa..(old contents) & i1 <=
52                    j & j < sa..(old csize)" */
53                /*: note "(j + 1, v2) : sa..contents & i1 + 1 <= j + 1 & j + 1 <
54                    sa..csize" */
55                /*: note "(EX i. (i, v2) : sa..contents & i1 + 1 <= i & i < sa..csize)"
56                    */
57            }
58            /*: note "(EX i. (i, v2) : sa..contents & i1 + 1 <= i & i < sa..csize)" */
59        }
60        int r2a = sa.index0f(v2);
61        /*: assume "r2a < 0 | (0 <= r2a & r2a < i1) | (r2a = i1 & (i1 + 1, v2) :
62            sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize)" */
63
64        int r2b = sb.index0f(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
55     v1, Object v2)
56 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
57     sa..contents = sb..contents & sa..csize = sb..csize"
58     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
59     "sb..msize"
60     ensures "True" */
61 {
62     /*: assume "~((EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
63     v2) | ((EX i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
64     sa..contents & i1 <= i & i < sa..csize) & v1 ~= v2))" */
65     /*: assume "0 <= i1 & i1 <= sa..csize" */
66     sa.add_at(i1, v1);
67     /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
68     int r2a = sa.lastIndexOf(v2);
69
70     int r2b = sb.lastIndexOf(v2);
71     /*: assume "0 <= i1 & i1 <= sb..csize" */
72     sb.add_at(i1, v1);
73
74     {
75         /*: assuming "(EX i. (i, v2) : sa..(old contents) & i1 <= i & i < sa..(old
76         csize))" */
77         /*: note "(r2b + 1, v2) : sa__contents" */
78     }
79
80     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
81     */
82 }
83
84 static void add_at_lastIndexOf_between_s_10(ArrayList sa, ArrayList sb, int i1,
85     Object v1, Object v2)
86 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
87     sa..contents = sb..contents & sa..csize = sb..csize"
88     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
89     "sb..msize"
90     ensures "True" */
91 {
92     /*: assume "0 <= i1 & i1 <= sa..csize" */
93     sa.add_at(i1, v1);
94     /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | ((EX i.
95     (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) : sa..contents &
96     i1 <= i & i < sa..csize))" */
97
98     {
99         /*: assuming "(EX i. (i, v2) : sa..(old contents) & i1 <= i & i < sa..(old
100         csize))" */
101         {
102             /*: pickWitness j :: int suchThat "(j, v2) : sa..(old contents) & i1 <=
103             j & j < sa..(old csize)" */
104             /*: note "(j + 1, v2) : sa..contents & i1 + 1 <= j + 1 & j + 1 <
105             sa..csize" */
106             /*: note "(EX i. (i, v2) : sa..contents & i1 + 1 <= i & i < sa..csize)"
107             */
108         }
109         /*: note "(EX i. (i, v2) : sa..contents & i1 + 1 <= i & i < sa..csize)" */
110     }
111
112     int r2a = sa.lastIndexOf(v2);
113
114     int r2b = sb.lastIndexOf(v2);
115     sb.add_at(i1, v1);

```

```

100     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
101 }
102
103
104 static void add_at_lastIndexOf_between_c_10(ArrayList sa, ArrayList sb, int i1,
105     Object v1, Object v2)
106 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
107     sa..contents = sb..contents & sa..csize = sb..csize"
108     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
109     "sb..msize"
110     ensures "True" */
111 {
112     /*: assume "0 <= i1 & i1 <= sa..csize" */
113     sa.add_at(i1, v1);
114     /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
115     /*: assume "~((EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | ((EX
116         i. (i, v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) :
117         sa..contents & i1 <= i & i < sa..csize)))" */
118     int r2a = sa.lastIndexOf(v2);
119
120     int r2b = sb.lastIndexOf(v2);
121     /*: assume "0 <= i1 & i1 <= sb..csize" */
122     sb.add_at(i1, v1);
123
124     {
125         /*: assuming "(EX i. (i, v2) : sa..(old contents) & i1 <= i & i < sa..(old
126             csize))" */
127         /*: note "(r2b + 1, v2) : sa__contents" */
128     }
129
130     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
131         */
132 }
133
134 static void add_at_lastIndexOf_post_s_11(ArrayList sa, ArrayList sb, int i1, Object
135     v1, Object v2)
136 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
137     sa..contents = sb..contents & sa..csize = sb..csize"
138     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
139     "sb..msize"
140     ensures "True" */
141 {
142     /*: assume "0 <= i1 & i1 <= sa..csize" */
143     sa.add_at(i1, v1);
144     {
145         /*: assuming "(EX i. (i, v2) : sa..(old contents) & i1 <= i & i < sa..(old
146             csize))" */
147         {
148             /*: pickWitness j :: int suchThat "(j, v2) : sa..(old contents) & i1 <=
149                 j & j < sa..(old csize)" */
150             /*: note "(j + 1, v2) : sa..contents & i1 + 1 <= j + 1 & j + 1 <
151                 sa..csize" */
152             /*: note "(EX i. (i, v2) : sa..contents & i1 + 1 <= i & i < sa..csize)"
153                 */
154         }
155         /*: note "(EX i. (i, v2) : sa..contents & i1 + 1 <= i & i < sa..csize)" */
156     }
157     int r2a = sa.lastIndexOf(v2);
158     /*: assume "r2a < 0 | (0 <= r2a & r2a < i1)" */
159
160     int r2b = sb.lastIndexOf(v2);
161     sb.add_at(i1, v1);
162
163     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
164 }

```



```

153
154 static void add_at_lastIndexOf_post_c_11(ArrayList sa, ArrayList sb, int i1, Object
155     v1, Object v2)
156 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
157     sa..contents = sb..contents & sa..csize = sb..csize"
158     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
159     "sb..msize"
160     ensures "True" */
161 {
162     /*: assume "0 <= i1 & i1 <= sa..csize" */
163     sa.add_at(i1, v1);
164     /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
165     int r2a = sa.lastIndexOf(v2);
166     /*: assume "~(r2a < 0 | (0 <= r2a & r2a < i1))" */
167
168     int r2b = sb.lastIndexOf(v2);
169     /*: assume "0 <= i1 & i1 <= sb..csize" */
170     sb.add_at(i1, v1);
171
172     {
173         /*: assuming "(EX i. (i, v2) : sa..(old contents) & i1 <= i & i < sa..(old
174             csize))" */
175         /*: note "(r2b + 1, v2) : sa__contents" */
176     }
177
178     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
179     */
180 }
181
182 static void add_at_remove_at_pre_c_15(ArrayList sa, ArrayList sb, int i1, Object
183     v1, int i2)
184 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
185     sa..contents = sb..contents & sa..csize = sb..csize"
186     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
187     "sb..msize"
188     ensures "True" */
189 {
190     /*: assume "~((i1 < i2 & i2 < sa..csize & (ALL v. ((i2 - 1, v) : sa..contents)
191         = ((i2, v) : sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 &
192         i2 < sa..csize) | (sa..csize - 1 >= i1 & i1 = i2 & (i1, v1) : sa..contents &
193         0 <= i1 & i1 < sa..csize) | (sa..csize - 1 >= i1 & i1 > i2 & (i1, v1) :
194         sa..contents & 0 <= i1 & i1 < sa..csize))" */
195     /*: assume "0 <= i1 & i1 <= sa..csize" */
196     sa.add_at(i1, v1);
197     /*: assume "0 <= i2 & i2 < sa..csize" */
198     sa.remove_at(i2);
199
200     /*: assume "0 <= i2 & i2 < sb..csize" */
201     sb.remove_at(i2);
202     /*: assume "0 <= i1 & i1 <= sb..csize" */
203     sb.add_at(i1, v1);
204
205     {
206         /*: assuming "i1 < i2 & ~(ALL v. ((i2 - 1, v) : sa..(old contents)) = ((i2,
207             v) : sa..(old contents)))" */
208         {
209             /*: pickWitness w :: obj suchThat "((i2 - 1, w) ~: sa..(old contents))
210                 & ((i2, w) : sa..(old contents))" */
211             /*: note "(i2, w) : sa..contents" */
212             /*: note "(i2, w) ~: sb..contents" */
213             /*: note "sa..contents ~= sb..contents" */
214         }
215         /*: note "sa..contents ~= sb..contents" */
216     }
217 }

```

```

206     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
207 }
208
209 static void add_at_remove_at_between_c_16(ArrayList sa, ArrayList sb, int i1,
    Object v1, int i2)
210 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
211     sa..contents = sb..contents & sa..csize = sb..csize"
212     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
    "sb..msize"
213     ensures "True" */
214 {
215     /*: assume "0 <= i1 & i1 <= sa..csize" */
216     sa.add_at(i1, v1);
217     /*: 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))" */
218     /*: assume "0 <= i2 & i2 < sa..csize" */
219     sa.remove_at(i2);
220
221     /*: assume "0 <= i2 & i2 < sb..csize" */
222     sb.remove_at(i2);
223     /*: assume "0 <= i1 & i1 <= sb..csize" */
224     sb.add_at(i1, v1);
225
226     {
227         /*: assuming "i1 < i2 & ~(ALL v. ((i2 - 1, v) : sa..(old contents)) = ((i2,
    v) : sa..(old contents)))" */
228         {
229             /*: pickWitness w :: obj suchThat "((i2 - 1, w) ~: sa..(old contents))
    & ((i2, w) : sa..(old contents))" */
230             /*: note "(i2, w) : sa..contents" */
231             /*: note "(i2, w) ~: sb..contents" */
232             /*: note "sa..contents ~= sb..contents" */
233         }
234         /*: note "sa..contents ~= sb..contents" */
235     }
236
237     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
238 }
239
240 static void add_at_set_pre_c_21(ArrayList sa, ArrayList sb, int i1, Object v1, int
    i2, Object v2)
241 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
242     sa..contents = sb..contents & sa..csize = sb..csize"
243     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
    "sb..msize"
244     ensures "True" */
245 {
246     /*: 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)" */
247     /*: assume "0 <= i1 & i1 <= sa..csize" */
248     sa.add_at(i1, v1);
249     /*: assume "0 <= i2 & i2 < sa..csize" */
250     sa.set(i2, v2);
251
252     /*: assume "0 <= i2 & i2 < sb..csize" */
253     sb.set(i2, v2);
254     /*: assume "0 <= i1 & i1 <= sb..csize" */
255     sb.add_at(i1, v1);
256

```

```

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.
274     (i, v1) : sa..contents & 0 <= i & i < i2) | ~(EX i. (i, v1) : sa..contents
275     & 0 <= i & i < i2) & (i2, v1) : sa..contents & 0 <= i2 & i2 < sa..csize & i2
276     < sa..csize - 1 & (i2 + 1, v1) : sa..contents & 0 <= i2 + 1 & i2 + 1 <
277     sa..csize)" */
278     int r1a = sa.indexOf(v1);
279     /*: assume "0 <= i2 & i2 < sa..csize" */
280     Object r2a = sa.remove_at(i2);
281
282     Object r2b = sb.remove_at(i2);
283     /*: note "~(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < i2) & (i2, v1) :
284     sa..(old contents) & 0 <= i2 & i2 < sa..(old csize) & i2 < sa..(old csize) -
285     1 & (i2 + 1, v1) : sa..(old contents) & 0 <= i2 + 1 & i2 + 1 < sa..(old
286     csize) --> (i2, v1) : sb..contents" */
287     int r1b = sb.indexOf(v1);
288
289     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
290     sb..csize" */
291 }
292
293 static void indexOf_remove_at_pre_c_66(ArrayList sa, ArrayList sb, Object v1, int
    i2)
294 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
295     sa..contents = sb..contents & sa..csize = sb..csize"
296 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
297 ensures "True" */
298 {
299     /*: assume "~(~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
300     (i, v1) : sa..contents & 0 <= i & i < i2) | ~(EX i. (i, v1) : sa..contents
301     & 0 <= i & i < i2) & (i2, v1) : sa..contents & 0 <= i2 & i2 < sa..csize & i2
302     < sa..csize - 1 & (i2 + 1, v1) : sa..contents & 0 <= i2 + 1 & i2 + 1 <
303     sa..csize))" */
304     int r1a = sa.indexOf(v1);
305     /*: assume "0 <= i2 & i2 < sa..csize" */
306     Object r2a = sa.remove_at(i2);
307
308     /*: assume "0 <= i2 & i2 < sb..csize" */
309     Object r2b = sb.remove_at(i2);
310     /*: ghost specvar sb__contents :: "(int * obj) set" = "sb..contents" */
311     int r1b = sb.indexOf(v1);
312
313     {
314         /*: assuming "(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < sa..(old
315         csize)) & ~(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < i2) & (i2,
316         v1) ~: sa..(old contents)" */
317         /*: note "(r1a - 1, v1) : sb__contents" */
318     }
319 }

```

```

306     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
307         sb..csize)" */
308 }
309 static void indexOf_remove_at_between_s_67(ArrayList sa, ArrayList sb, Object v1,
310     int i2)
311 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
312     sa..contents = sb..contents & sa..csize = sb..csize"
313     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
314     ensures "True" */
315 {
316     int r1a = sa.indexOf(v1);
317     /*: assume "~(r1a < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize - 1 &
318         (i2 + 1, v1) : sa..contents & 0 <= i2 + 1 & i2 + 1 < sa..csize)" */
319     /*: assume "0 <= i2 & i2 < sa..csize" */
320     Object r2a = sa.remove_at(i2);
321     Object r2b = sb.remove_at(i2);
322     /*: note "~(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < i2) & (i2, v1) :
323         sa..(old contents) & 0 <= i2 & i2 < sa..(old csize) & i2 < sa..(old csize) -
324         1 & (i2 + 1, v1) : sa..(old contents) & 0 <= i2 + 1 & i2 + 1 < sa..(old
325         csize)) --> (i2, v1) : sb..contents" */
326     int r1b = sb.indexOf(v1);
327     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
328         sb..csize" */
329 }
330 static void indexOf_remove_at_between_c_67(ArrayList sa, ArrayList sb, Object v1,
331     int i2)
332 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
333     sa..contents = sb..contents & sa..csize = sb..csize"
334     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
335     ensures "True" */
336 {
337     int r1a = sa.indexOf(v1);
338     /*: assume "~(r1a < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize - 1
339         & (i2 + 1, v1) : sa..contents & 0 <= i2 + 1 & i2 + 1 < sa..csize))" */
340     /*: assume "0 <= i2 & i2 < sa..csize" */
341     Object r2a = sa.remove_at(i2);
342     /*: assume "0 <= i2 & i2 < sb..csize" */
343     Object r2b = sb.remove_at(i2);
344     /*: ghost specvar sb__contents :: "(int * obj) set" = "sb..contents" */
345     int r1b = sb.indexOf(v1);
346     {
347         /*: assuming "(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < sa..(old
348             csize)) & ~(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < i2) & (i2,
349             v1) ~: sa..(old contents)" */
350         /*: note "(r1a - 1, v1) : sb__contents" */
351     }
352     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
353         sb..csize)" */
354 }
355 static void indexOf_remove_at_post_s_68(ArrayList sa, ArrayList sb, Object v1, int
356     i2)
357 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
358     sa..contents = sb..contents & sa..csize = sb..csize"
359     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
360     ensures "True" */
361 {
362     int r1a = sa.indexOf(v1);

```

```

358     /*: assume "0 <= i2 & i2 < sa..csize" */
359     Object r2a = sa.remove_at(i2);
360     /*: assume "r1a < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize & (i2,
361         v1) : sa..contents & 0 <= i2 & i2 < sa..csize)" */
362
363     Object r2b = sb.remove_at(i2);
364     /*: note "(~(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < i2) & (i2, v1) :
365         sa..(old contents) & 0 <= i2 & i2 < sa..(old csize) & i2 < sa..(old csize) -
366         1 & (i2 + 1, v1) : sa..(old contents) & 0 <= i2 + 1 & i2 + 1 < sa..(old
367         csize)) --> (i2, v1) : sb..contents" */
368     int r1b = sb.indexOf(v1);
369
370     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
371         sb..csize" */
372 }
373
374 static void indexOf_remove_at_pre_s_69(ArrayList sa, ArrayList sb, Object v1, int
375     i2)
376 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
377     sa..contents = sb..contents & sa..csize = sb..csize"
378 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
379 ensures "True" */
380 {
381     /*: assume "~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
382         (i, v1) : sa..contents & 0 <= i & i < i2) | (~(EX i. (i, v1) : sa..contents
383         & 0 <= i & i < i2) & (i2, v1) : sa..contents & 0 <= i2 & i2 < sa..csize & i2
384         < sa..csize - 1 & (i2 + 1, v1) : sa..contents & 0 <= i2 + 1 & i2 + 1 <
385         sa..csize)" */
386     int r1a = sa.indexOf(v1);
387     /*: assume "0 <= i2 & i2 < sa..csize" */
388     sa.remove_at(i2);
389
390     sb.remove_at(i2);
391     /*: note "(~(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < i2) & (i2, v1) :
392         sa..(old contents) & 0 <= i2 & i2 < sa..(old csize) & i2 < sa..(old csize) -
393         1 & (i2 + 1, v1) : sa..(old contents) & 0 <= i2 + 1 & i2 + 1 < sa..(old
394         csize)) --> (i2, v1) : sb..contents" */
395     int r1b = sb.indexOf(v1);
396
397     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
398 }
399
400 static void indexOf_remove_at_pre_c_69(ArrayList sa, ArrayList sb, Object v1, int
401     i2)
402 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
403     sa..contents = sb..contents & sa..csize = sb..csize"
404 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
405 ensures "True" */
406 {
407     /*: assume "~(~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
408         (i, v1) : sa..contents & 0 <= i & i < i2) | (~(EX i. (i, v1) : sa..contents
409         & 0 <= i & i < i2) & (i2, v1) : sa..contents & 0 <= i2 & i2 < sa..csize & i2
410         < sa..csize - 1 & (i2 + 1, v1) : sa..contents & 0 <= i2 + 1 & i2 + 1 <
411         sa..csize))" */
412     int r1a = sa.indexOf(v1);
413     /*: assume "0 <= i2 & i2 < sa..csize" */
414     sa.remove_at(i2);
415
416     /*: assume "0 <= i2 & i2 < sb..csize" */
417     sb.remove_at(i2);
418     /*: ghost specvar sb__contents :: "(int * obj) set" = "sb..contents" */
419     int r1b = sb.indexOf(v1);
420
421     {
422

```

```

404     /*: assuming "(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < sa..(old
405         csize)) & ~(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < i2) & (i2,
406         v1) ~: sa..(old contents)" */
407     /*: note "(r1a - 1, v1) : sb__contents" */
408 }
409
410     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
411     */
412 }
413
414 static void indexOf_remove_at_between_s_70(ArrayList sa, ArrayList sb, Object v1,
415     int i2)
416 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
417     sa..contents = sb..contents & sa..csize = sb..csize"
418     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
419     ensures "True" */
420 {
421     int r1a = sa.indexOf(v1);
422     /*: assume "(r1a < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize - 1 &
423         (i2 + 1, v1) : sa..contents & 0 <= i2 + 1 & i2 + 1 < sa..csize))" */
424     /*: assume "0 <= i2 & i2 < sa..csize" */
425     sa.remove_at(i2);
426
427     sb.remove_at(i2);
428     /*: note "~(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < i2) & (i2, v1) :
429         sa..(old contents) & 0 <= i2 & i2 < sa..(old csize) & i2 < sa..(old csize) -
430         1 & (i2 + 1, v1) : sa..(old contents) & 0 <= i2 + 1 & i2 + 1 < sa..(old
431         csize)) --> (i2, v1) : sb..contents" */
432     int r1b = sb.indexOf(v1);
433
434     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
435 }
436
437 static void indexOf_remove_at_between_c_70(ArrayList sa, ArrayList sb, Object v1,
438     int i2)
439 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
440     sa..contents = sb..contents & sa..csize = sb..csize"
441     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
442     ensures "True" */
443 {
444     int r1a = sa.indexOf(v1);
445     /*: assume "~(r1a < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize - 1
446         & (i2 + 1, v1) : sa..contents & 0 <= i2 + 1 & i2 + 1 < sa..csize))" */
447     /*: assume "0 <= i2 & i2 < sa..csize" */
448     sa.remove_at(i2);
449
450     /*: assume "0 <= i2 & i2 < sb..csize" */
451     sb.remove_at(i2);
452     /*: ghost specvar sb__contents :: "(int * obj) set" = "sb..contents" */
453     int r1b = sb.indexOf(v1);
454
455     {
456         /*: assuming "(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < sa..(old
457             csize)) & ~(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < i2) & (i2,
458             v1) ~: sa..(old contents)" */
459         /*: note "(r1a - 1, v1) : sb__contents" */
460     }
461
462     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
463     */
464 }
465
466 static void indexOf_remove_at_post_s_71(ArrayList sa, ArrayList sb, Object v1, int
467     i2)
468 /*: 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 "(ria < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize & (i2,
463         v1) : sa..contents & 0 <= i2 & i2 < sa..csize))" */
464     sb.remove_at(i2);
465     /*: note "(~(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < i2) & (i2, v1) :
466         sa..(old contents) & 0 <= i2 & i2 < sa..(old csize) & i2 < sa..(old csize) -
467         1 & (i2 + 1, v1) : sa..(old contents) & 0 <= i2 + 1 & i2 + 1 < sa..(old
468         csize)) --> (i2, v1) : sb..contents" */
469     int r1b = sb.indexOf(v1);
470     /*: assert "r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize" */
471 }
472
473 static void indexOf_remove_at_post_c_71(ArrayList sa, ArrayList sb, Object v1, int
474     i2)
475 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
476     sa..contents = sb..contents & sa..csize = sb..csize"
477 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
478 ensures "True" */
479 {
480     int r1a = sa.indexOf(v1);
481     /*: assume "0 <= i2 & i2 < sa..csize" */
482     sa.remove_at(i2);
483     /*: assume "(ria < 0 | (0 <= r1a & r1a < i2) | (r1a = i2 & i2 < sa..csize &
484         (i2, v1) : sa..contents & 0 <= i2 & i2 < sa..csize))" */
485     /*: assume "0 <= i2 & i2 < sb..csize" */
486     sb.remove_at(i2);
487     /*: ghost specvar sb__contents :: "(int * obj) set" = "sb..contents" */
488     int r1b = sb.indexOf(v1);
489     {
490         /*: assuming "(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < sa..(old
491             csize)) & ~(EX i. (i, v1) : sa..(old contents) & 0 <= i & i < i2) & (i2,
492             v1) ~: sa..(old contents)" */
493         /*: note "(r1a - 1, v1) : sb__contents" */
494     }
495     /*: assert "(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
496         */
497 }
498
499 static void lastIndexOf_add_at_pre_c_81(ArrayList sa, ArrayList sb, Object v1, int
500     i2, Object v2)
501 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
502     sa..contents = sb..contents & sa..csize = sb..csize"
503 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
504     "sb..msize"
505 ensures "True" */
506 {
507     /*: assume "(~(EX i. (i, v1) : sa..contents & 0 <= i & i < sa..csize) & v1 ~=
508         v2) | ((EX i. (i, v1) : sa..contents & 0 <= i & i < i2) & ~(EX i. (i, v1) :
509         sa..contents & i2 <= i & i < sa..csize) & v1 ~= v2))" */
510     int r1a = sa.lastIndexOf(v1);
511     /*: assume "0 <= i2 & i2 <= sa..csize" */
512     sa.add_at(i2, v2);
513     /*: 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
513             csizer))" */
514         /*: note "(r1a + 1, v1) : sb__contents" */
515     }
516
517     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csizer = sb..csizer)"
518         */
519 }
520
521 static void lastIndexOf_add_at_between_c_82(ArrayList sa, ArrayList sb, Object v1,
522     int i2, Object v2)
523 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
524     sa..contents = sb..contents & sa..csizer = sb..csizer"
525     modifies "sa..contents", "sb..contents", "sa..csizer", "sb..csizer", "sa..msizer",
526     "sb..msizer"
527     ensures "True" */
528 {
529     int r1a = sa.lastIndexOf(v1);
530     /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2))" */
531     /*: assume "0 <= i2 & i2 <= sa..csizer" */
532     sa.add_at(i2, v2);
533
534     /*: assume "0 <= i2 & i2 <= sb..csizer" */
535     sb.add_at(i2, v2);
536     /*: ghost specvar sb__contents :: "(int * obj) set" = "sb..contents" */
537     int r1b = sb.lastIndexOf(v1);
538
539     {
540         /*: assuming "(EX i. (i, v1) : sa..(old contents) & i2 <= i & i < sa..(old
541             csizer))" */
542         /*: note "(r1a + 1, v1) : sb__contents" */
543     }
544
545     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csizer = sb..csizer)"
546         */
547 }
548
549 static void lastIndexOf_add_at_post_c_83(ArrayList sa, ArrayList sb, Object v1, int
550     i2, Object v2)
551 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
552     sa..contents = sb..contents & sa..csizer = sb..csizer"
553     modifies "sa..contents", "sb..contents", "sa..csizer", "sb..csizer", "sa..msizer",
554     "sb..msizer"
555     ensures "True" */
556 {
557     int r1a = sa.lastIndexOf(v1);
558     /*: assume "0 <= i2 & i2 <= sa..csizer" */
559     sa.add_at(i2, v2);
560     /*: assume "~((r1a < 0 & v1 ~= v2) | (0 <= r1a & r1a < i2 & v1 ~= v2))" */
561
562     /*: assume "0 <= i2 & i2 <= sb..csizer" */
563     sb.add_at(i2, v2);
564     /*: ghost specvar sb__contents :: "(int * obj) set" = "sb..contents" */
565     int r1b = sb.lastIndexOf(v1);
566
567     {
568         /*: assuming "(EX i. (i, v1) : sa..(old contents) & i2 <= i & i < sa..(old
569             csizer))" */
570         /*: note "(r1a + 1, v1) : sb__contents" */
571     }
572 }

```



```

563     /*: assert "~(r1a = r1b & sa..contents = sb..contents & sa..csize = sb..csize)"
564     */
565 }
566
567 static void remove_at_index0f_pre_s_114(ArrayList sa, ArrayList sb, int i1, Object
568     v2)
569 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
570     sa..contents = sb..contents & sa..csize = sb..csize"
571     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
572     ensures "True" */
573 {
574     /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
575     (i, v2) : sa..contents & 0 <= i & i < i1) | ~(EX i. (i, v2) : sa..contents
576     & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & i1
577     < sa..csize - 1 & (i1 + 1, v2) : sa..contents & 0 <= i1 + 1 & i1 + 1 <
578     sa..csize)" */
579     /*: assume "0 <= i1 & i1 < sa..csize" */
580     Object r1a = sa.remove_at(i1);
581     /*: note "~(EX i. (i, v2) : sa..(old contents) & 0 <= i & i < i1) & (i1, v2) :
582     sa..(old contents) & 0 <= i1 & i1 < sa..(old csize) & i1 < sa..(old csize) -
583     1 & (i1 + 1, v2) : sa..(old contents) & 0 <= i1 + 1 & i1 + 1 < sa..(old
584     csize)) --> (i1, v2) : sa..contents" */
585     int r2a = sa.index0f(v2);
586
587     int r2b = sb.index0f(v2);
588     Object r1b = sb.remove_at(i1);
589
590     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
591     sb..csize" */
592 }
593
594 static void remove_at_index0f_pre_c_114(ArrayList sa, ArrayList sb, int i1, Object
595     v2)
596 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
597     sa..contents = sb..contents & sa..csize = sb..csize"
598     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
599     ensures "True" */
600 {
601     /*: assume "~(~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) | (EX i.
602     (i, v2) : sa..contents & 0 <= i & i < i1) | ~(EX i. (i, v2) : sa..contents
603     & 0 <= i & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & i1
604     < sa..csize - 1 & (i1 + 1, v2) : sa..contents & 0 <= i1 + 1 & i1 + 1 <
605     sa..csize))" */
606     /*: assume "0 <= i1 & i1 < sa..csize" */
607     Object r1a = sa.remove_at(i1);
608     /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
609     int r2a = sa.index0f(v2);
610
611     int r2b = sb.index0f(v2);
612     /*: assume "0 <= i1 & i1 < sb..csize" */
613     Object r1b = sb.remove_at(i1);
614
615     {
616         /*: assuming "(EX i. (i, v2) : sa..(old contents) & 0 <= i & i < sa..(old
617         csize)) & ~(EX i. (i, v2) : sa..(old contents) & 0 <= i & i < i1) & (i1,
618         v2) ~: sa..(old contents)" */
619         /*: note "(r2b - 1, v2) : sa__contents" */
620     }
621
622     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
623     sb..csize)" */
624 }

```

```

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

```

```

659 static void remove_at_indexOf_post_s_116(ArrayList sa, ArrayList sb, int i1, Object
660     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"
664     ensures "True" */
665 {
666     /*: assume "0 <= i1 & i1 < sa..csize" */
667     Object r1a = sa.remove_at(i1);
668     {
669         /*: assuming "(EX i. (i, v2) : sa..(old contents) & i1 + 1 <= i & i <
670             sa..(old csize))" */
671         {
672             /*: pickWitness j :: int suchThat "(j, v2) : sa..(old contents) & i1 +
673                 1 <= j & j < sa..(old csize)" */
674             /*: note "(j - 1, v2) : sa..contents & i1 <= j - 1 & j - 1 < sa..csize"
675                 */
676             /*: note "(EX i. (i, v2) : sa..contents & i1 <= i & i < sa..csize)" */
677         }
678         /*: note "(EX i. (i, v2) : sa..contents & i1 <= i & i < sa..csize)" */
679     }
680     int r2a = sa.indexOf(v2);
681     /*: assume "(r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1) | (r2a = i1 & r1a =
682         v2 & i1 < sa..csize)" */
683
684     int r2b = sb.indexOf(v2);
685     Object r1b = sb.remove_at(i1);
686
687     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
688         sb..csize" */
689 }
690
691 static void remove_at_indexOf_post_c_116(ArrayList sa, ArrayList sb, int i1, Object
692     v2)
693 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
694     sa..contents = sb..contents & sa..csize = sb..csize"
695     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
696     ensures "True" */
697 {
698     /*: assume "0 <= i1 & i1 < sa..csize" */
699     Object r1a = sa.remove_at(i1);
700     /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
701     int r2a = sa.indexOf(v2);
702     /*: assume "~((r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1) | (r2a = i1 & r1a =
703         v2 & i1 < sa..csize))" */
704
705     int r2b = sb.indexOf(v2);
706     /*: assume "0 <= i1 & i1 < sb..csize" */
707     Object r1b = sb.remove_at(i1);
708
709     {
710         /*: assuming "(EX i. (i, v2) : sa..(old contents) & 0 <= i & i < sa..(old
711             csize)) & ~(EX i. (i, v2) : sa..(old contents) & 0 <= i & i < i1) & (i1,
712             v2) ~: sa..(old contents)" */
713         /*: note "(r2b - 1, v2) : sa__contents" */
714     }
715
716     /*: assert "~(r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
717         sb..csize)" */
718 }
719
720 static void remove_at_lastIndexOf_between_s_118(ArrayList sa, ArrayList sb, int i1,
721     Object v2)
722 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
723     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 &
       sa..contents = sb..contents & sa..csize = sb..csize"
737 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
738 ensures "True" */
739 {
740     /*: assume "0 <= i1 & i1 < sa..csize" */
741     Object r1a = sa.remove_at(i1);
742     {
743         /*: assuming "(EX i. (i, v2) : sa..(old contents) & i1 + 1 <= i & i <
           sa..(old csize))" */
744         {
745             /*: pickWitness j :: int suchThat "(j, v2) : sa..(old contents) & i1 +
               1 <= j & j < sa..(old csize)" */
746             /*: note "(j - 1, v2) : sa..contents & i1 <= j - 1 & j - 1 < sa..csize"
               */
747             /*: note "(EX i. (i, v2) : sa..contents & i1 <= i & i < sa..csize)" */
748         }
749         /*: note "(EX i. (i, v2) : sa..contents & i1 <= i & i < sa..csize)" */
750     }
751     int r2a = sa.lastIndexOf(v2);
752     /*: assume "(r2a < 0 & r1a ~= v2) | (0 <= r2a & r2a < i1 & r1a ~= v2)" */
753
754     int r2b = sb.lastIndexOf(v2);
755     Object r1b = sb.remove_at(i1);
756
757     /*: assert "r1a = r1b & r2a = r2b & sa..contents = sb..contents & sa..csize =
       sb..csize" */
758 }
759
760 static void remove_at_set_pre_c_129(ArrayList sa, ArrayList sb, int i1, int i2,
       Object v2)
761 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
       sa..contents = sb..contents & sa..csize = sb..csize"
762 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
763
764

```

```

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..csize", "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" */

```

```

866 }
867
868 static void remove_at_indexOf_pre_c_141(ArrayList sa, ArrayList sb, int i1, Object
      v2)
869 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
870             sa..contents = sb..contents & sa..csize = sb..csize"
871             modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
872             ensures "True" */
873 {
874     /*: 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))" */
875     /*: assume "0 <= i1 & i1 < sa..csize" */
876     sa.remove_at(i1);
877     /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
878     int r2a = sa.indexOf(v2);
879
880     int r2b = sb.indexOf(v2);
881     /*: assume "0 <= i1 & i1 < sb..csize" */
882     sb.remove_at(i1);
883
884     {
885         /*: 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)" */
886         /*: note "(r2b - 1, v2) : sa__contents" */
887     }
888
889     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
      */
890 }
891
892 static void remove_at_indexOf_between_s_142(ArrayList sa, ArrayList sb, int i1,
      Object v2)
893 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
894             sa..contents = sb..contents & sa..csize = sb..csize"
895             modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
896             ensures "True" */
897 {
898     /*: assume "0 <= i1 & i1 < sa..csize" */
899     sa.remove_at(i1);
900     /*: 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)" */
901     {
902         /*: assuming "(EX i. (i, v2) : sa..(old contents) & i1 + 1 <= i & i <
          sa..(old csize))" */
903         {
904             /*: pickWitness j :: int suchThat "(j, v2) : sa..(old contents) & i1 +
              1 <= j & j < sa..(old csize)" */
905             /*: note "(j - 1, v2) : sa..contents & i1 <= j - 1 & j - 1 < sa..csize"
              */
906             /*: note "(EX i. (i, v2) : sa..contents & i1 <= i & i < sa..csize)" */
907         }
908         /*: note "(EX i. (i, v2) : sa..contents & i1 <= i & i < sa..csize)" */
909     }
910     int r2a = sa.indexOf(v2);
911
912     int r2b = sb.indexOf(v2);
913     sb.remove_at(i1);
914

```

```

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,
919     Object v2)
920 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
921     sa..contents = sb..contents & sa..csize = sb..csize"
922 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
923 ensures "True" */
924 {
925     /*: assume "0 <= i1 & i1 < sa..csize" */
926     sa.remove_at(i1);
927     /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
928     /*: assume "~((~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & (i1,
929     v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | (EX i. (i, v2)
930     : sa..contents & 0 <= i & i < i1) | ~(EX i. (i, v2) : sa..contents & 0 <= i
931     & i < i1) & (i1, v2) : sa..contents & 0 <= i1 & i1 < sa..csize & (i1, v2) :
932     sa..(old contents) & 0 <= i1 & i1 < sa..(old csize) & i1 < sa..csize))" */
933     int r2a = sa.indexOf(v2);
934
935     int r2b = sb.indexOf(v2);
936     /*: assume "0 <= i1 & i1 < sb..csize" */
937     sb.remove_at(i1);
938
939     {
940         /*: assuming "(EX i. (i, v2) : sa..(old contents) & 0 <= i & i < sa..(old
941         csize)) & ~(EX i. (i, v2) : sa..(old contents) & 0 <= i & i < i1) & (i1,
942         v2) ~: sa..(old contents)" */
943         /*: note "(r2b - 1, v2) : sa__contents" */
944     }
945
946     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
947     */
948 }
949
950 static void remove_at_indexOf_post_s_143(ArrayList sa, ArrayList sb, int i1, Object
951     v2)
952 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
953     sa..contents = sb..contents & sa..csize = sb..csize"
954 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
955 ensures "True" */
956 {
957     /*: assume "0 <= i1 & i1 < sa..csize" */
958     sa.remove_at(i1);
959     {
960         /*: assuming "(EX i. (i, v2) : sa..(old contents) & i1 + 1 <= i & i <
961         sa..(old csize))" */
962         {
963             /*: pickWitness j :: int suchThat "(j, v2) : sa..(old contents) & i1 +
964             1 <= j & j < sa..(old csize)" */
965             /*: note "(j - 1, v2) : sa..contents & i1 <= j - 1 & j - 1 < sa..csize"
966             */
967             /*: note "(EX i. (i, v2) : sa..contents & i1 <= i & i < sa..csize)" */
968         }
969         /*: note "(EX i. (i, v2) : sa..contents & i1 <= i & i < sa..csize)" */
970     }
971     int r2a = sa.indexOf(v2);
972     /*: assume "(r2a < 0 & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old
973     csize)) | (0 <= r2a & r2a < i1) | (r2a = i1 & (i1, v2) : sa..(old contents)
974     & 0 <= i1 & i1 < sa..(old csize) & i1 < sa..csize)" */
975
976     int r2b = sb.indexOf(v2);
977     sb.remove_at(i1);
978
979     /*: 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
969     v2)
970 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
971     sa..contents = sb..contents & sa..csize = sb..csize"
972     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
973     ensures "True" */
974 {
975     /*: assume "0 <= i1 & i1 < sa..csize" */
976     sa.remove_at(i1);
977     /*: ghost specvar sa__contents :: "(int * obj) set" = "sa..contents" */
978     int r2a = sa.indexOf(v2);
979     /*: assume "~((r2a < 0 & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 <
980     sa..(old csize)) | (0 <= r2a & r2a < i1) | (r2a = i1 & (i1, v2) : sa..(old
981     contents) & 0 <= i1 & i1 < sa..(old csize) & i1 < sa..csize))" */
982     int r2b = sb.indexOf(v2);
983     /*: assume "0 <= i1 & i1 < sb..csize" */
984     sb.remove_at(i1);
985
986     {
987         /*: assuming "(EX i. (i, v2) : sa..(old contents) & 0 <= i & i < sa..(old
988         csize)) & ~(EX i. (i, v2) : sa..(old contents) & 0 <= i & i < i1) & (i1,
989         v2) ~: sa..(old contents)" */
990         /*: note "(r2b - 1, v2) : sa__contents" */
991     }
992
993     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
994     */
995 }
996
997 static void remove_at_lastIndexOf_between_s_145(ArrayList sa, ArrayList sb, int i1,
998     Object v2)
999 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1000     sa..contents = sb..contents & sa..csize = sb..csize"
1001     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1002     ensures "True" */
1003 {
1004     /*: assume "0 <= i1 & i1 < sa..csize" */
1005     sa.remove_at(i1);
1006     /*: assume "~(EX i. (i, v2) : sa..contents & 0 <= i & i < sa..csize) & (i1,
1007     v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old csize)) | ((EX i. (i,
1008     v2) : sa..contents & 0 <= i & i < i1) & ~(EX i. (i, v2) : sa..contents & i1
1009     <= i & i < sa..csize) & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 <
1010     sa..(old csize))" */
1011
1012     {
1013         /*: assuming "(EX i. (i, v2) : sa..(old contents) & i1 + 1 <= i & i <
1014         sa..(old csize))" */
1015         {
1016             /*: pickWitness j :: int suchThat "(j, v2) : sa..(old contents) & i1 +
1017             1 <= j & j < sa..(old csize)" */
1018             /*: note "(j - 1, v2) : sa..contents & i1 <= j - 1 & j - 1 < sa..csize"
1019             */
1020             /*: note "(EX i. (i, v2) : sa..contents & i1 <= i & i < sa..csize)" */
1021         }
1022         /*: note "(EX i. (i, v2) : sa..contents & i1 <= i & i < sa..csize)" */
1023     }
1024     int r2a = sa.lastIndexOf(v2);
1025
1026     int r2b = sb.lastIndexOf(v2);
1027     sb.remove_at(i1);
1028
1029     /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
1030 }

```

```

1017 static void remove_at_lastIndexOf_post_s_146(ArrayList sa, ArrayList sb, int i1,
1018         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 <
1028             sa..(old csize))" */
1029         {
1030             /*: pickWitness j :: int suchThat "(j, v2) : sa..(old contents) & i1 +
1031                 1 <= j & j < sa..(old csize)" */
1032             /*: note "(j - 1, v2) : sa..contents & i1 <= j - 1 & j - 1 < sa..csize"
1033                 */
1034             /*: note "(EX i. (i, v2) : sa..contents & i1 <= i & i < sa..csize)" */
1035             }
1036             /*: note "(EX i. (i, v2) : sa..contents & i1 <= i & i < sa..csize)" */
1037         }
1038         int r2a = sa.lastIndexOf(v2);
1039         /*: assume "(r2a < 0 & (i1, v2) ~: sa..(old contents) & 0 <= i1 & i1 < sa..(old
1040             csize) | (0 <= r2a & r2a < i1 & (i1, v2) ~: sa..(old contents) & 0 <= i1 &
1041             i1 < sa..(old csize))" */
1042
1043         int r2b = sb.lastIndexOf(v2);
1044         sb.remove_at(i1);
1045
1046         /*: assert "r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize" */
1047     }
1048
1049 static void remove_at_remove_at_pre_c_147(ArrayList sa, ArrayList sb, int i1, int
1050     i2)
1051 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1052             sa..contents = sb..contents & sa..csize = sb..csize"
1053     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1054     ensures "True" */
1055 {
1056     /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
1057         sa..contents)) & 0 <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
1058         sa..csize) | (i1 = i2 & (ALL v. ((i1, v) : sa..contents) = ((i2 + 1, v) :
1059         sa..contents)) & 0 <= i1 & i1 < sa..csize & 0 <= i2 + 1 & i2 + 1 <
1060         sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
1061         sa..contents) = ((i1 + 1, v) : sa..contents)) & 0 <= i1 & i1 < sa..csize & 0
1062         <= i1 + 1 & i1 + 1 < sa..csize))" */
1063     /*: assume "0 <= i1 & i1 < sa..csize" */
1064     sa.remove_at(i1);
1065     /*: assume "0 <= i2 & i2 < sa..csize" */
1066     Object r2a = sa.remove_at(i2);
1067
1068     /*: assume "0 <= i2 & i2 < sb..csize" */
1069     Object r2b = sb.remove_at(i2);
1070     /*: assume "0 <= i1 & i1 < sb..csize" */
1071     sb.remove_at(i1);
1072
1073     {
1074         /*: assuming "i1 > i2 & ~(ALL v. ((i1, v) : sa..(old contents)) = ((i1 + 1,
1075             v) : sa..(old contents)))" */
1076         {
1077             /*: pickWitness w :: obj suchThat "((i1, w) : sa..(old contents)) &
1078                 ((i1 + 1, w) ~: sa..(old contents))" */
1079             /*: note "(i1 - 1, w) ~: sa..contents" */
1080             /*: note "(i1 - 1, w) : sb..contents" */
1081         }
1082     }

```

```

1067         /*: note "sa..contents ~= sb..contents" */
1068     }
1069     /*: note "sa..contents ~= sb..contents" */
1070 }
1071
1072     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
1073 }
1074
1075 static void remove_at_remove_at_between_c_148(ArrayList sa, ArrayList sb, int i1,
        int i2)
1076 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1077     sa..contents = sb..contents & sa..csize = sb..csize"
1078 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1079 ensures "True" */
1080 {
1081     /*: assume "0 <= i1 & i1 < sa..csize" */
1082     sa.remove_at(i1);
1083     /*: 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))" */
1084     /*: assume "0 <= i2 & i2 < sa..csize" */
1085     Object r2a = sa.remove_at(i2);
1086
1087     /*: assume "0 <= i2 & i2 < sb..csize" */
1088     Object r2b = sb.remove_at(i2);
1089     /*: assume "0 <= i1 & i1 < sb..csize" */
1090     sb.remove_at(i1);
1091
1092     {
1093         /*: assuming "i1 > i2 & ~(ALL v. ((i1, v) : sa..(old contents)) = ((i1 + 1,
            v) : sa..(old contents)))" */
1094         {
1095             /*: pickWitness w :: obj suchThat "((i1, w) : sa..(old contents)) &
                ((i1 + 1, w) ~: sa..(old contents))" */
1096             /*: note "(i1 - 1, w) ~: sa..contents" */
1097             /*: note "(i1 - 1, w) : sb..contents" */
1098             /*: note "sa..contents ~= sb..contents" */
1099         }
1100         /*: note "sa..contents ~= sb..contents" */
1101     }
1102
1103     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
        */
1104 }
1105
1106 static void remove_at_remove_at_pre_c_150(ArrayList sa, ArrayList sb, int i1, int
        i2)
1107 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1108     sa..contents = sb..contents & sa..csize = sb..csize"
1109 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1110 ensures "True" */
1111 {
1112     /*: 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))" */
1113     /*: assume "0 <= i1 & i1 < sa..csize" */
1114     sa.remove_at(i1);
1115     /*: assume "0 <= i2 & i2 < sa..csize" */

```

```

1116     sa.remove_at(i2);
1117
1118     /*: assume "0 <= i2 & i2 < sb..csize" */
1119     sb.remove_at(i2);
1120     /*: assume "0 <= i1 & i1 < sb..csize" */
1121     sb.remove_at(i1);
1122
1123     {
1124         /*: assuming "i1 < i2 & ~(ALL v. ((i2, v) : sa..(old contents)) = ((i2 + 1,
1125             v) : sa..(old contents)))" */
1126         {
1127             /*: pickWitness w :: obj suchThat "((i2, w) : sa..(old contents)) &
1128                 ((i2 + 1, w) ~: sa..(old contents))" */
1129             /*: note "(i2 - 1, w) : sa..contents" */
1130             /*: note "(i2 - 1, w) ~: sb..contents" */
1131             /*: note "sa..contents ~= sb..contents" */
1132         }
1133         /*: note "sa..contents ~= sb..contents" */
1134     }
1135     {
1136         /*: assuming "i1 > i2 & ~(ALL v. ((i1, v) : sa..(old contents)) = ((i1 + 1,
1137             v) : sa..(old contents)))" */
1138         {
1139             /*: pickWitness w :: obj suchThat "((i1, w) : sa..(old contents)) &
1140                 ((i1 + 1, w) ~: sa..(old contents))" */
1141             /*: note "(i1 - 1, w) ~: sa..contents" */
1142             /*: note "(i1 - 1, w) : sb..contents" */
1143             /*: note "sa..contents ~= sb..contents" */
1144         }
1145         /*: note "sa..contents ~= sb..contents" */
1146     }
1147
1148     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
1149 }
1150
1151 static void remove_at_remove_at_between_c_151(ArrayList sa, ArrayList sb, int i1,
1152     int i2)
1153 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1154     sa..contents = sb..contents & sa..csize = sb..csize"
1155 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1156 ensures "True" */
1157 {
1158     /*: assume "0 <= i1 & i1 < sa..csize" */
1159     sa.remove_at(i1);
1160     /*: assume "~((i1 < i2 & (ALL v. ((i2 - 1, v) : sa..contents) = ((i2, v) :
1161         sa..contents)) & 0 <= i2 - 1 & i2 - 1 < sa..csize & 0 <= i2 & i2 <
1162         sa..csize) | i1 = i2 | (sa..csize > i1 & i1 > i2 & (ALL v. ((i1, v) :
1163         sa..(old contents)) = ((i1, v) : sa..contents)) & 0 <= i1 & i1 < sa..(old
1164         csize) & 0 <= i1 & i1 < sa..csize))" */
1165     /*: assume "0 <= i2 & i2 < sa..csize" */
1166     sa.remove_at(i2);
1167
1168     /*: assume "0 <= i2 & i2 < sb..csize" */
1169     sb.remove_at(i2);
1170     /*: assume "0 <= i1 & i1 < sb..csize" */
1171     sb.remove_at(i1);
1172
1173     {
1174         /*: assuming "i1 < i2 & ~(ALL v. ((i2, v) : sa..(old contents)) = ((i2 + 1,
1175             v) : sa..(old contents)))" */
1176         {
1177             /*: pickWitness w :: obj suchThat "((i2, w) : sa..(old contents)) &
1178                 ((i2 + 1, w) ~: sa..(old contents))" */
1179             /*: note "(i2 - 1, w) : sa..contents" */
1180             /*: note "(i2 - 1, w) ~: sb..contents" */

```

```

1170         /*: note "sa..contents ~= sb..contents" */
1171     }
1172     /*: note "sa..contents ~= sb..contents" */
1173 }
1174 {
1175     /*: assuming "i1 > i2 & ~(ALL v. ((i1, v) : sa..(old contents)) = ((i1 + 1,
1176     v) : sa..(old contents)))" */
1177     {
1178         /*: pickWitness w :: obj suchThat "((i1, w) : sa..(old contents)) &
1179         ((i1 + 1, w) ~: sa..(old contents))" */
1178         /*: note "(i1 - 1, w) ~: sa..contents" */
1179         /*: note "(i1 - 1, w) : sb..contents" */
1180         /*: note "sa..contents ~= sb..contents" */
1181     }
1182     /*: note "sa..contents ~= sb..contents" */
1183 }
1184
1185     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
1186 }
1187
1188 static void remove_at_set_pre_c_156(ArrayList sa, ArrayList sb, int i1, int i2,
1189     Object v2)
1189 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1190     sa..contents = sb..contents & sa..csize = sb..csize"
1191     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1192     ensures "True" */
1193 {
1194     /*: assume "~((i1 < i2 & (ALL v. ((i2, v) : sa..contents) = ((i2 + 1, v) :
1195     sa..contents)) & (i2, v2) : sa..contents & (i2 + 1, v2) : sa..contents & 0
1196     <= i2 & i2 < sa..csize & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 = i2 & (i2
1197     + 1, v2) : sa..contents & 0 <= i2 + 1 & i2 + 1 < sa..csize) | (i1 > i2))" */
1198     /*: assume "0 <= i1 & i1 < sa..csize" */
1199     sa.remove_at(i1);
1200     /*: assume "0 <= i2 & i2 < sa..csize" */
1201     sa.set(i2, v2);
1202
1203     /*: assume "0 <= i2 & i2 < sb..csize" */
1204     sb.set(i2, v2);
1205     /*: assume "0 <= i1 & i1 < sb..csize" */
1206     sb.remove_at(i1);
1207
1208     {
1209         /*: assuming "i1 < i2 & (i2, v2) ~: sa..(old contents)" */
1210         /*: note "(i2 - 1, v2) ~: sa..contents" */
1211         /*: note "(i2 - 1, v2) : sb..contents" */
1212         /*: note "sa..contents ~= sb..contents" */
1213     }
1214
1215     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
1216 }
1217
1218 static void set_add_at_pre_c_189(ArrayList sa, ArrayList sb, int i1, Object v1, int
1219     i2, Object v2)
1220 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1221     sa..contents = sb..contents & sa..csize = sb..csize"
1222     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
1223     "sb..msize"
1224     ensures "True" */
1225 {
1226     /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..contents & (i1, v2) :
1227     sa..contents & v1 = v2 & 0 <= i1 & i1 < sa..csize) | (i1 > i2 & (ALL v. ((i1
1228     - 1, v) : sa..contents) = ((i1, v) : sa..contents)) & (i1 - 1, v1) :
1229     sa..contents & (i1, v1) : sa..contents & 0 <= i1 - 1 & i1 - 1 < sa..csize &
1230     0 <= i1 & i1 < sa..csize))" */
1231     /*: assume "0 <= i1 & i1 < sa..csize" */

```

```

1223     sa.set(i1, v1);
1224     /*: assume "0 <= i2 & i2 <= sa..csize" */
1225     sa.add_at(i2, v2);
1226
1227     /*: assume "0 <= i2 & i2 <= sb..csize" */
1228     sb.add_at(i2, v2);
1229     /*: assume "0 <= i1 & i1 < sb..csize" */
1230     sb.set(i1, v1);
1231
1232     {
1233         /*: assuming "i1 >= i2 & (i1, v1) ~: sa..(old contents)" */
1234         /*: note "(i1 + 1, v1) : sa..contents" */
1235         /*: note "sa..contents ~= sb..contents" */
1236     }
1237
1238     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
1239 }
1240
1241 static void set_add_at_between_c_190(ArrayList sa, ArrayList sb, int i1, Object v1,
1242     int i2, Object v2)
1243 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1244     sa..contents = sb..contents & sa..csize = sb..csize"
1245     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
1246     "sb..msize"
1247     ensures "True" */
1248 {
1249     /*: assume "0 <= i1 & i1 < sa..csize" */
1250     sa.set(i1, v1);
1251     /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & (i1, v2) :
1252     sa..(old contents) & v1 = v2 & 0 <= i1 & i1 < sa..(old csize)) | (i1 > i2 &
1253     (ALL v. ((i1 - 1, v) : sa..contents) = ((i1, v) : sa..(old contents))) & (i1
1254     - 1, v1) : sa..contents & (i1, v1) : sa..(old contents) & 0 <= i1 - 1 & i1 -
1255     1 < sa..csize & 0 <= i1 & i1 < sa..(old csize)))" */
1256     /*: assume "0 <= i2 & i2 <= sa..csize" */
1257     sa.add_at(i2, v2);
1258
1259     /*: assume "0 <= i2 & i2 <= sb..csize" */
1260     sb.add_at(i2, v2);
1261     /*: assume "0 <= i1 & i1 < sb..csize" */
1262     sb.set(i1, v1);
1263
1264     {
1265         /*: assuming "i1 >= i2 & (i1, v1) ~: sa..(old contents)" */
1266         /*: note "(i1 + 1, v1) : sa..contents" */
1267         /*: note "sa..contents ~= sb..contents" */
1268     }
1269
1270     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
1271 }
1272
1273 static void set_add_at_post_c_191(ArrayList sa, ArrayList sb, int i1, Object v1,
1274     int i2, Object v2)
1275 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1276     sa..contents = sb..contents & sa..csize = sb..csize"
1277     modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize", "sa..msize",
1278     "sb..msize"
1279     ensures "True" */
1280 {
1281     /*: assume "0 <= i1 & i1 < sa..csize" */
1282     sa.set(i1, v1);
1283     /*: assume "0 <= i2 & i2 <= sa..csize" */
1284     sa.add_at(i2, v2);
1285     /*: assume "~(i1 < i2 | (i1 = i2 & (i1, v1) : sa..(old contents) & (i1, v2) :
1286     sa..(old contents) & v1 = v2 & 0 <= i1 & i1 < sa..(old csize)) | (i1 > i2 &
1287     (ALL v. ((i1, v) : sa..contents) = ((i1, v) : sa..(old contents))) & (i1, v1)

```

```

1278         : sa..contents & (i1, v1) : sa..(old contents) & 0 <= i1 & i1 < sa..csize &
1279         0 <= i1 & i1 < sa..(old csize)))" */
1280
1281     /*: assume "0 <= i2 & i2 <= sb..csize" */
1282     sb.add_at(i2, v2);
1283     /*: assume "0 <= i1 & i1 < sb..csize" */
1284     sb.set(i1, v1);
1285
1286     {
1287         /*: assuming "i1 >= i2 & (i1, v1) ~: sa..(old contents)" */
1288         /*: note "(i1 + 1, v1) : sa..contents" */
1289         /*: note "sa..contents ~= sb..contents" */
1290     }
1291
1292     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
1293 }
1294
1295 static void set_remove_at_pre_c_201(ArrayList sa, ArrayList sb, int i1, Object v1,
1296 int i2)
1297 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1298 sa..contents = sb..contents & sa..csize = sb..csize"
1299 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1300 ensures "True" */
1301 {
1302     /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (ALL v. ((i1, v) :
1303 sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
1304 (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
1305 1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v. ((i1, v) :
1306 sa..contents) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..contents &
1307 (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..csize & 0 <= i1 + 1 & i1 +
1308 1 < sa..csize))" */
1309     /*: assume "0 <= i1 & i1 < sa..csize" */
1310     sa.set(i1, v1);
1311     /*: assume "0 <= i2 & i2 < sa..csize" */
1312     Object r2a = sa.remove_at(i2);
1313
1314     /*: assume "0 <= i2 & i2 < sb..csize" */
1315     Object r2b = sb.remove_at(i2);
1316     /*: assume "0 <= i1 & i1 < sb..csize" */
1317     sb.set(i1, v1);
1318
1319     {
1320         /*: assuming "i1 > i2 & (i1, v1) ~: sa..(old contents)" */
1321         /*: note "(i1 - 1, v1) : sa..contents" */
1322         /*: note "sa..contents ~= sb..contents" */
1323     }
1324
1325     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
1326 */
1327 }
1328
1329 static void set_remove_at_between_c_202(ArrayList sa, ArrayList sb, int i1, Object
1330 v1, int i2)
1331 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1332 sa..contents = sb..contents & sa..csize = sb..csize"
1333 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1334 ensures "True" */
1335 {
1336     /*: assume "0 <= i1 & i1 < sa..csize" */
1337     sa.set(i1, v1);
1338     /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (ALL v. ((i1, v) :
1339 sa..(old contents)) = ((i1 + 1, v) : sa..contents)) & (i1, v1) : sa..(old
1340 contents) & (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) & 0
1341 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1 > i2 & (ALL v.
1342 ((i1, v) : sa..(old contents)) = ((i1 + 1, v) : sa..contents)) & (i1, v1) :

```

```

1328     sa..(old contents) & (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old
1329     csize) & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
1330     /*: assume "0 <= i2 & i2 < sa..csize" */
1331     Object r2a = sa.remove_at(i2);
1332     /*: assume "0 <= i2 & i2 < sb..csize" */
1333     Object r2b = sb.remove_at(i2);
1334     /*: assume "0 <= i1 & i1 < sb..csize" */
1335     sb.set(i1, v1);
1336
1337     {
1338         /*: assuming "i1 > i2 & (i1, v1) ~: sa..(old contents)" */
1339         /*: note "(i1 - 1, v1) : sa..contents" */
1340         /*: note "sa..contents ~= sb..contents" */
1341     }
1342
1343     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
1344     */
1345 }
1346
1347 static void set_remove_at_post_c_203(ArrayList sa, ArrayList sb, int i1, Object v1,
1348 int i2)
1349 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1350 sa..contents = sb..contents & sa..csize = sb..csize"
1351 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1352 ensures "True" */
1353 {
1354     /*: assume "0 <= i1 & i1 < sa..csize" */
1355     sa.set(i1, v1);
1356     /*: assume "0 <= i2 & i2 < sa..csize" */
1357     Object r2a = sa.remove_at(i2);
1358     /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize & (ALL v. ((i1, v) : sa..(old
1359     contents)) = ((i1, v) : sa..contents)) & (i1, v1) : sa..(old contents) &
1360     (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 & i1 <
1361     sa..csize) | (sa..csize > i1 & i1 > i2 & (ALL v. ((i1, v) : sa..(old
1362     contents)) = ((i1, v) : sa..contents)) & (i1, v1) : sa..(old contents) &
1363     (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 & i1 <
1364     sa..csize))" */
1365
1366     /*: assume "0 <= i2 & i2 < sb..csize" */
1367     Object r2b = sb.remove_at(i2);
1368     /*: assume "0 <= i1 & i1 < sb..csize" */
1369     sb.set(i1, v1);
1370
1371     {
1372         /*: assuming "i1 > i2 & (i1, v1) ~: sa..(old contents)" */
1373         /*: note "(i1 - 1, v1) : sa..contents" */
1374         /*: note "sa..contents ~= sb..contents" */
1375     }
1376
1377     /*: assert "~(r2a = r2b & sa..contents = sb..contents & sa..csize = sb..csize)"
1378     */
1379 }
1380
1381 static void set_remove_at_pre_c_204(ArrayList sa, ArrayList sb, int i1, Object v1,
1382 int i2)
1383 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1384 sa..contents = sb..contents & sa..csize = sb..csize"
1385 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1386 ensures "True" */
1387 {
1388     /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, v1) :
1389     sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
1390     > i2 & (ALL v. ((i1, v) : sa..contents) = ((i1 + 1, v) : sa..contents)) &

```



```

1378         (i1, v1) : sa..contents & (i1 + 1, v1) : sa..contents & 0 <= i1 & i1 <
1379         sa..csize & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
1380     /*: assume "0 <= i1 & i1 < sa..csize" */
1381     sa.set(i1, v1);
1382     /*: assume "0 <= i2 & i2 < sa..csize" */
1383     sa.remove_at(i2);
1384
1385     /*: assume "0 <= i2 & i2 < sb..csize" */
1386     sb.remove_at(i2);
1387     /*: assume "0 <= i1 & i1 < sb..csize" */
1388     sb.set(i1, v1);
1389
1390     {
1391         /*: assuming "i1 > i2 & (i1, v1) ~: sa..(old contents)" */
1392         /*: note "(i1 - 1, v1) : sa..contents" */
1393         /*: note "sa..contents ~= sb..contents" */
1394     }
1395
1396     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
1397 }
1398
1399 static void set_remove_at_between_c_205(ArrayList sa, ArrayList sb, int i1, Object
1400 v1, int i2)
1401 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1402 sa..contents = sb..contents & sa..csize = sb..csize"
1403 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1404 ensures "True" */
1405 {
1406     /*: assume "0 <= i1 & i1 < sa..csize" */
1407     sa.set(i1, v1);
1408     /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize - 1 & (i1 + 1, v1) :
1409 sa..contents & 0 <= i1 + 1 & i1 + 1 < sa..csize) | (sa..csize - 1 > i1 & i1
1410 > i2 & (ALL v. ((i1, v) : sa..(old contents)) = ((i1 + 1, v) :
1411 sa..contents)) & (i1, v1) : sa..(old contents) & (i1 + 1, v1) : sa..contents
1412 & 0 <= i1 & i1 < sa..(old csize) & 0 <= i1 + 1 & i1 + 1 < sa..csize))" */
1413     /*: assume "0 <= i2 & i2 < sa..csize" */
1414     sa.remove_at(i2);
1415
1416     /*: assume "0 <= i2 & i2 < sb..csize" */
1417     sb.remove_at(i2);
1418     /*: assume "0 <= i1 & i1 < sb..csize" */
1419     sb.set(i1, v1);
1420
1421     {
1422         /*: assuming "i1 > i2 & (i1, v1) ~: sa..(old contents)" */
1423         /*: note "(i1 - 1, v1) : sa..contents" */
1424         /*: note "sa..contents ~= sb..contents" */
1425     }
1426
1427     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
1428 }
1429
1430 static void set_remove_at_post_c_206(ArrayList sa, ArrayList sb, int i1, Object v1,
1431 int i2)
1432 /*: requires "sa ~= null & sb ~= null & sa ~= sb & sa..init & sb..init &
1433 sa..contents = sb..contents & sa..csize = sb..csize"
1434 modifies "sa..contents", "sb..contents", "sa..csize", "sb..csize"
1435 ensures "True" */
1436 {
1437     /*: assume "0 <= i1 & i1 < sa..csize" */
1438     sa.set(i1, v1);
1439     /*: assume "0 <= i2 & i2 < sa..csize" */
1440     sa.remove_at(i2);
1441     /*: assume "~(i1 < i2 | (i1 = i2 & i2 < sa..csize & (i1, v1) : sa..contents & 0
1442 <= i1 & i1 < sa..csize) | (sa..csize > i1 & i1 > i2 & (ALL v. ((i1, v) :

```

```

1434     sa..(old contents)) = ((i1, v) : sa..(contents)) & (i1, v1) : sa..(old
1435     contents) & (i1, v1) : sa..contents & 0 <= i1 & i1 < sa..(old csize) & 0 <=
1436     i1 & i1 < sa..csize))" */
1437
1438     /*: assume "0 <= i2 & i2 < sb..csize" */
1439     sb.remove_at(i2);
1440     /*: assume "0 <= i1 & i1 < sb..csize" */
1441     sb.set(i1, v1);
1442
1443     {
1444         /*: assuming "i1 > i2 & (i1, v1) ~: sa..(old contents)" */
1445         /*: note "(i1 - 1, v1) : sa..contents" */
1446         /*: note "sa..contents ~= sb..contents" */
1447     }
1448
1449     /*: assert "~(sa..contents = sb..contents & sa..csize = sb..csize)" */
1450 }

```

### B.6.3 Inverse Testing Methods

Listing 20. ArrayListInv.java

```

1 class ArrayListInv {
2     static void add_at_0(ArrayList s, int i, Object v)
3     /*: requires "s ~= null & s..init & 0 <= i & i <= s..csize"
4     modifies "s..contents", "s..csize", "s..msize"
5     ensures "True" */
6     {
7         s.add_at(i, v);
8         s.remove_at(i);
9
10        /*: assert "s..contents = s..(old contents) & s..csize = s..(old csize)" */
11    }
12
13    static void remove_at_1(ArrayList s, int i)
14    /*: requires "s ~= null & s..init & 0 <= i & i < s..csize"
15    modifies "s..contents", "s..csize", "s..msize"
16    ensures "True" */
17    {
18        Object r = s.remove_at(i);
19        s.add_at(i, r);
20
21        /*: assert "s..contents = s..(old contents) & s..csize = s..(old csize)" */
22    }
23
24    static void set_2(ArrayList s, int i, Object v)
25    /*: requires "s ~= null & s..init & 0 <= i & i < s..csize"
26    modifies "s..contents"
27    ensures "True" */
28    {
29        Object r = s.set(i, v);
30        s.set(i, r);
31
32        /*: assert "s..contents = s..(old contents) & s..csize = s..(old csize)" */
33    }
34 }
35

```

