8-1-2003

Questions from the February 2003 CSIS Programming Competition

CSIS Faculty
Pace University

SCHOOL OF COMPUTER SCIENCE AND INFORMATION SYSTEMS

TECHNICAL REPORT

Number 192, August 2003

Questions from the February 2003 CSIS Programming Competition

CSIS Faculty

PACE UNIVERSITY

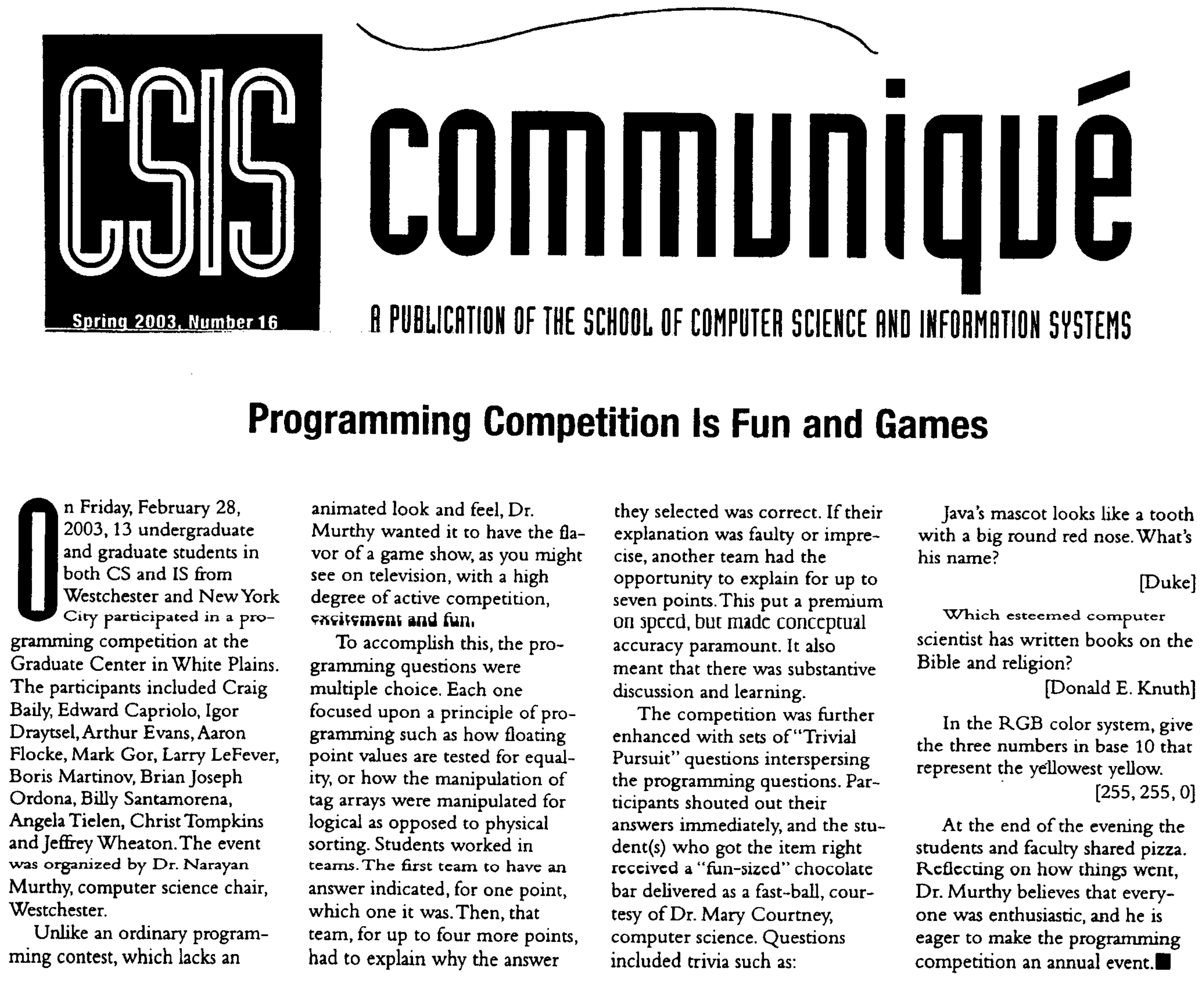
The current issue of Technical Reports presents the questions used at the CSIS Programming Competition on February 28, 2003 (and their answers).

Dr. Narayan Murthy, Chairperson of the Computer Science Department in Westchester, conceived of this event, designed its format, and organized it.

Many faculty members, including full-timers and adjuncts in Computer Science and Information Systems, contributed questions.

Drs. Christelle Scharff and Anthony Joseph, from the Computer Science Department in New York, and Dr. Narayan Murthy served as judges.

The piece describing this event, from the Spring 2003 issue of the CSIS Communiqué (Number 16), is reproduced below.



CSIS communiqué

Spring 2003, Number 16

A PUBLICATION OF THE SCHOOL OF COMPUTER SCIENCE AND INFORMATION SYSTEMS

Programming Competition Is Fun and Games

On Friday, February 28, 2003, 13 undergraduate and graduate students in both CS and IS from Westchester and New York City participated in a programming competition at the Graduate Center in White Plains. The participants included Craig Baily, Edward Capriolo, Igor Draytsel, Arthur Evans, Aaron Flocke, Mark Gor, Larry LeFever, Boris Martinov, Brian Joseph Ordona, Billy Santamorena, Angela Tielen, Christ Tompkins and Jeffrey Wheaton. The event was organized by Dr. Narayan Murthy, computer science chair, Westchester.

Unlike an ordinary programming contest, which lacks an animated look and feel, Dr. Murthy wanted it to have the flavor of a game show, as you might see on television, with a high degree of active competition, excitement and fun.

To accomplish this, the programming questions were multiple choice. Each one focused upon a principle of programming such as how floating point values are tested for equality, or how the manipulation of tag arrays were manipulated for logical as opposed to physical sorting. Students worked in teams. The first team to have an answer indicated, for one point, which one it was. Then, that team, for up to four more points, had to explain why the answer they selected was correct. If their explanation was faulty or imprecise, another team had the opportunity to explain for up to seven points. This put a premium on speed, but made conceptual accuracy paramount. It also meant that there was substantive discussion and learning.

The competition was further enhanced with sets of "Trivial Pursuit" questions interspersing the programming questions. Participants shouted out their answers immediately, and the student(s) who got the item right received a "fun-sized" chocolate bar delivered as a fast-ball, courtesy of Dr. Mary Courtney, computer science. Questions included trivia such as:

Java's mascot looks like a tooth with a big round red nose. What's his name?

[Duke]

Which esteemed computer scientist has written books on the Bible and religion?

[Donald E. Knuth]

In the RGB color system, give the three numbers in base 10 that represent the yellowest yellow.

[255, 255, 0]

At the end of the evening the students and faculty shared pizza. Reflecting on how things went, Dr. Murthy believes that everyone was enthusiastic, and he is eager to make the programming competition an annual event.■

Multiple-Choice Programming Questions
(with Answers)

The following questions were posed to the competing teams of students. They were pitched to be accessible to students having the equivalent of a year of programming in Java.

For each question, the first team with an answer raised its hand. If the answer was correct, the team received one point and had an opportunity to give an explanation for up to four more points. If their explanation was faulty or imprecise, another team had the chance to present the reason for the answer for up to seven points. Points were awarded by an panel of three judges. This put a premium on speed but made conceptual accuracy paramount. It also meant there was substantive discussion and learning.

The order of the items here differs from the order in which they were presented.

1. What is displayed by the following fragment of code?

```
int accumulator = 0;

int count = 0;

while (count < 10)
  {
     accumulator = 1 - accumulator;

     count = count + 1;
  }

System.out.println("accumulator = " + accumulator);
```

ANSWERS:

a) 0

b) -8

c) -9

d) There is no output because the **while** loop is infinite.

Correct answer: a - The value of **accumulator** following pass 1 and following every odd-numbered pass thereafter is 1; the value of **accumulator** following each even-numbered pass is 0.

Because the **while** loop iterates ten times the value displayed for **accumulator** is 0.

2. The following code is intended to exchange the values stored by the **int** variables **a** and **b**, but the code does not work.

```
a = b;   //a gets the value of b
b = a;   //b gets the value of a
```

Which of the methods below could be used as a Java facility for swapping?

```
public class Swapper
  {
    public static void swapOne(int a, int b)
      {
        int holder;
        holder = a;
        a = b;
        b = holder;
      }

   public static void swapTwo(int a, int b)
      {
        a = a + b;
        b = a - b;
        a = a - b;
      }
  }
```

ANSWERS:

a) **swapOne()** but not **swapTwo()**

b) **swapTwo()** but not **swapOne()**

c) Both **swapOne()** and **swapTwo()**

d) Neither **swapOne()** nor **swapTwo()**

Correct answer: d - because arguments are passed by value

3. Suppose that **x1** and **x2** are **floats**. **x1** gets the value of f(t), and **x2** holds the value of g(t), where t represents a time in milliseconds stored by a **long**.

```
float  x1 = 0f,  x2 = 0f;

//code omitted

while (currentTime < endTime)
  {
    x1 =  f(currentTime);
    x2 =  g(currentTime);

    if ( x1 == x2 )
      {
        System.out.println("Alert at time " + currentTime))
      }

    //code omitted
  }
```

This could may work fine for years, but one day misperform by failing to flag **x1** and **x2** as equal when, mathematically, they are. Why?

ANSWERS: a) because there may be errors in the omitted code

b) because, to quote a truism pertaining to real arithmetic: "10 times .1 is hardly ever 1"

c) because **floats** have less precision than **doubles** (the repair is to declare **x1** and **x2** as **double**)

d) because **x1** and **x2** are superfluous and should not be used. The loop should open like this:

```
while (currentTime < endTime)
  {
     // x1 = f(currentTime);
     // x2 = g(currentTime);
     //
     // if ( x1 == x2 )

     if ( f(currentTime) == g(currentTime) )
       {
         etc.
```

Correct answer: b - Real arithmetic grinds away precision. Two results that are arithmetically equivalent may not compute to exactly the same value.

To test **doubles** and **floats** for equality, code like this, where .00001 stands for the epsilon of your choice:

```
if ( Math.abs(x1 - x2)  <  .00001 )
```

4. One of the first applications of computers was to generate numerical solutions of differential equations. The technique was to express the differential equation as a difference equation and use a loop in which "the time t+1 value" was computed from "the time t value," just like computing the accumulation of interest in savings account.

For example, to solve the differential equation representing the cooling of a cup of coffee:

$$\frac{d\ \text{coffeeTemp}}{dt} = \text{constant} * (\text{coffeeTemp} - \text{roomTemp})$$

the expression below would be iterated, and the successive values of **nextCoffeeTemp** given as the solution.

```
nextCoffeeTemp = currentCoffeeTemp -
                  constant * (currentCoffeeTemp - roomTemp);
```

The constant represents the rate with which heat is conducted away from whatever is holding the coffee in an *ad hoc* time unit. A china cup has a higher conductivity than a thermous.

Which of the following could be a correct implementation within the loop:

```
System.out.println(coffeeTemp);

while ( coffeeTemp - roomTemp  >  0.5 )  // 1/2 a degree
  {
    // answer goes here

    System.out.println( coffeeTemp );
  }
```

ANSWERS:

a)
```
int nextCoffeeTemp = coffeeTemp -
                      constant * (coffeeTemp - roomTemp);

coffeeTemp = nextCoffeeTemp;
```

b)
```
coffeeTemp = coffeeTemp -
                  constant * (coffeeTemp - roomTemp);
```

c)
```
coffeeTemp -= constant * (coffeeTemp - roomTemp);
```

d) a and b

e) b and c

f) all of the above

g) none of the above

Correct answer: f

5. What is the output from the following attempt to add "one plus one." (Recall that the ASCII/Unicode value of '1' is 49, and the ASCII/Unicode value of '2' is 50.)

```
System.out.println( 1+1 + "   shows that   1 and 1 is 2" );

System.out.println("1 plus 1 equals " +  1+1 );
```

ANSWES:

a)
```
98   shows that   1 and 1 is 2
1 plus 1 equals 98
```

b)
```
2   shows that   1 and 1 is 2
1 plus 1 equals 2
```

c)
```
2   shows that   1 and 1 is 2
1 plus 1 equals 11
```

d)
```
11   shows that   1 and 1 is 2
1 plus 1 equals 11
```

e)
```
11  shows that  1 and 1 is 2
1 plus 1 equals 50
```

Correct Answer: c - The expression comprising the argument to **println()** is evaluated left to right. When the + operator's left operand or its right operand is a **String**, it performs concatenation; not addition.

In the upper statement, the leftmost + lies between two **ints**, therefore addition is performed.

In the lower statement, the leftmost + lies between a **String** and an **int**, therefore concatenation is performed, with an implicit **String.valueOf()** operation first applied to the **int**. This results in a **String**, which becomes the left operand for the next + operator.

Did anyone catch the grammatical error, repeated six times? The fix to the to make the verb agree with the plural subject: "1 and 1 are 2." Not, "1 and 1 is 2."

An special prize for anyone who can, in a grammatically correct way, write the following sentence using the standard 26 letters of the alphabet: "There are three *TÜ*s in the English language: to, too, and two."

6. Re-write the following method with the recursion removed:

```
void bunchOfNames(String name, int totalToShow, int soFarDisplayed)
 {
    if (soFarDisplayed < totalToShow)
      {
         System.out.println(name);

         soFarDisplayed++;

         bunchOfNames(name, totalToShow, soFarDisplayed);
      }
 }
```

ANSWERS:

a) trick question -- "tail recursion" cannot be removed

b)
```
void bunchOfNames(String name, int totalToShow, int soFarDisplayed)
 {
    for (int i = soFarDisplayed; i < totalToShow; i++)
      {
        System.out.println(name);
      }
 }
```

c)
```
void bunchOfNames(String name, int totalToShow, int soFarDisplayed)
 {
    for (int i = totalToShow; i > soFarDisplayed; i--)
      {
        System.out.println(name);
      }
 }
```

d)
```
void bunchOfNames(String name, int totalToShow)
 {
    for(int i = 0; i < totalToShow; i++)
      {
        System.out.println(name);
      }
 }
```

e) b and c

f) An event controlled loop (e.g. a **while** or a **do while** loop) is needed, not a count controlled loop (i.e. not a **for** loop).

g) Only an ignoramus would have written the given method recursively!

Correct Answer: e

d is unacceptable because it changes the API. Replace the kludgey recursive method with it, and preexiting calls will no longer run. Also, d assumes the user always kicks-off the recursive method with **soFarDisplayed** set to 0. This is a perfectly reasonable assumption, but an assumption nevertheless. Despite our valid reasoning, it may not always be true.

But, being practical, although **soFarDisplayed** is required as part of the recursive scaffolding it is not needed to capture information about the task from the user. (It really <u>should</u> start as zero.) One variable, the "number to print," (i.e. **totalToShow**) is conceptually sufficient.

Very often recursive methods require an an argument as a facilitating variable. When they do, a more reasonable interface should be offered in the form of a method that launches the recursive method:

```
void bunchOfNames(String name, int totalToShow)
  {
     bunchOfNames(name, totalToShow, 0);
  }
```

7. The utility for rounding shown below was written by a student in the algorithms course:

```
public class Rounder
  {
     public int round(double real)
       {
          int intPart = 0;
          intPart = (int) (real + 0.5);
          return intPart;
       }
  }
```

Of the following critiques, which two are most valid?

i) The code should be corrected to perform properly for negative arguments.

ii) The code should be corrected to perfrom properly for an argument of 0.

iii) A **long** should be returned, not an **int**.

iv) The method should be **static**.

v) There is no need for the local variable **intPart** -- just return the trucated sum.

ANSWERS:

a) iii and v

b) i and ii

c) i and iv

d) iii and iv

Correct Answer: c

Correctness is the most important attribute for any module or component; thus i is essential:

-3.1 should round to -3 and -3.6 should round to -4

The correction is a logical structure such as:

```
if (real >= 0)
     intPart  =  (int) (real + 0.5);
else
     intPart  =  (int) (real - 0.5);
```

Relative to object-oriented design, methods that merely process the value of an actual argument should be **static**. Objects are needed only when state has to be retained from call to call; that is, when something needs to be remembered in order for a successive call to work properly (such as a call on a **StringTokenizer** object).

If a Java aficionado needed to round, she would probably use one of the overloaded methods supplied by the **Math** class:
public static int round(float a) or **public static long round(double a)**

8. The following is the Fibonacci series. Its first and second terms are defined as 1 and 1 respectively. Thereafter, each successive term is the sum of the previous two.

```
leftCursor   rightCursor
        ↓     ↓
        1     1    2    3    5    8    13    21    34    55    ...
```

The fragment of code below is intended to displays the first fifteen terms of the Fibonacci series along a line, but it is not complete. What is missing, and where should it go?

```
                     int leftCursor  = 1;  //first term
                     int rightCursor = 1;  //second term

                     System.out.print("1 1 ");

                     for (int term = 2; term <= 15; term++)
                       {
point one --->        ........
                         int currentTerm = leftCursor + rightCursor;
point two --->        ........
                         System.out.print(currentTerm + " ");
point three --->      ........
                       }
```

ANSWERS:

a) at point one:
```
leftCursor  = rightCursor;
rightCursor = currentTerm;
```

b) at point two:
```
leftCursor  = rightCursor;
rightCursor = currentTerm;
```

c) at point two:
```
rightCursor = currentTerm;
leftCursor  = rightCursor;
```

d) at point three:
```
rightCursor = currentTerm;
leftCursor  = rightCursor;
```

Correct Answer: b

9. In 1671 the Scottish mathematician James Gregory discovered that pi could be expressed as the sum of the following convergent series.

pi = 4 * (1/1 - 1/3 + 1/5 - 1/7 + 1/9 - 1/11 ...)

This series is asethetic, but it converges slowly, making a vast number of terms needed for many places of precision. The following segment of code runs for two minutes:

```
int    minutes   = 2;   //minutes for program to run

long   durationInMilliseconds = minutes * 60 * 1000;
long   currentTime = System.currentTimeMillis();
long   endTime     = currentTime + durationInMilliseonds;

long   termsUsed = 0;
double sum = 0;
double demominator = 1;
while (System.currentTimeMillis() < endTime)
  {
    sum = sum + 1/denominator;
    denominator = denominator + 2;

    sum = sum - 1/denominator;
    denominator = denominator + 2;

    termsUsed = termsUsed + 2;
    System.out.println("pi with "+termsUsed+" terms is "+(4*sum));
  }
```

9.a. If you watch this code run, you can see how the sum settles down to pi from right to left. What one change would improve the code's performance more than any other, so that precision builds faster?

ANSWERS:

a) display intermediate results fewer times, such as only every thousand or ten thousand terms; for example:

```
if (termsUsed % 10000 == 0)
   System.out.println("pi with "+termsUsed+" terms is "+(4*sum));
```

b) eliminate the call to **System.currentTimeMillis()** from the test controlling the loop because executing the set-up and return from a method activation entails substantial overhead

c) recode the increments within the loop as shown below to decrease by five the number of memory fetches per interation:

```
sum += 1/denominator;
denominator +=  2;

sum -= 1/denominator;
denominator +=  2;

termsUsed += 2;
System.out.println("pi with "+termsUsed+" terms is "+(4*sum));
```

d) both Java and modern microprocessors are optimized to the point where performace is improved not by "diddling the code" but by finding a better algorithm (e.g. a more quickly convergent series)

Correct Answer: a - output is exceeding expensive, even to to the display screen; and to make matters worse, each of them here entails a "real" multiplicition

9.b. Suppose you wanted to compute pi to 100 places. Approximately how long would this code need to run?

ANSWERS:

a) This cannot be determined from the information available within the question.

b) Trick question: this code cannot compute pi to 100 places.

c) The answer is dependent upon the speed of the microprocessor but can be determined empirically. Because there are no nested loops, the code executes with an O(n) time complexity. Thus, for whatever time units it takes to compute n places (e.g. 20 places), it will take 100/n time units to compute 100 places.

d) The answer is dependent upon the speed of the microprocessor and can be determined empirically. However, even though the code performs linearly, the rate with which Gregory's function approaches its asymptotic limit is not. Thus, the empirical determination, while possible, is not something so easily accomplished.

Correct Answer: b - 100 places of precision cannot be stored by a **double**. The best possible approximation this code is capable of generating will have around 18 places.

10. **Math.random()** returns a double within the range of 0..1, excluding 1 (e.g. a value such as 0.00019383039804, 0.723322408398, or 0.998108991123). The values are drawn at random from a uniform distribution.

To get a random integer between 0 and 9, inclusive, one might code a statement such as the following:

```
int r = (int) (Math.random() * 10);
```

But, what would we code to get a random integer between 1 and n, inclusive; such as between 1 and 52?

ANSWERS:

a) **r = (int) (Math.random() * n) + 1;**

b) **r = (int) (Math.random() * (n+1));**

c) **r = (int) ((Math.random()+1) * n);**

d) **r = (int) (Math.random() * (n+1)) + 1;**

Correct Answer: a - So, how can this general idea be used to deal cards at random from a deck?

Put the 52 cards into an array; deal the card at **a[r]**, replacing it with the card at the end of the array; get another **r**, this time within a range decremented by 1; deal the card at **a[r]**, and continue.

```
Card[] a = new Card[52];

//copy the cards into array a

for (int i = 0; i < 52; i++)
  {
     int r = (int)(Math.random() * (52-i));  //r's range is 0..51

     deal a[r];

     a[r] = a[51 - i];
  }
```

11.a. Given a 100 element array named **data** in which the comparments at subscript 0 through subscript 65 hold "live data." Which fragment of code will move the block of data at subscript 10 through 65 "up one element"? We want the object currently in element 10 to be moved into element 11; the object currently in element 11 to be moved into element 12; and so on until the object currently in element 65 is moved into element 66.

ANSWERS:

```
a)  for (int i = 10;  i <= 65;  i++)   data[i+1] = data[i];

b)  for (int i = 10;  i <= 65;  i++)   data[i]   = data[i-1];

c)  for (int i = 65;  i >= 10;  i--)   data[i+1] = data[i];

d)  for (int i = 65;  i >= 10;  i--)   data[i]   = data[i-1];

e)  none of the above
```

Correct Answer: c - This has the effect of "opening compartment" **data[10]** so that it an incoming object can be stored there. An operation of this kind is needed for an insertion sort or for maintaining a priority queue in an array.

11.b. Given a 100 element array named **data** in which the comparments at subscript 0 through subscript 65 hold "live data" in sorted order. Which fragment of code will move the block of data at subscript 11 through 65 "down one element"? We want the object currently in element 65 to be moved into element 64; the object currently in element 64 to be moved into element 63; and so on until the value currently in element 11 is moved into element 10.

ANSWERS:

```
a)  for (int i = 11;  i <= 65;  i++)   data[i-1] = data[i];

b)  for (int i = 11;  i <= 65;  i++)   data[i]   = data[i-1];

c)  for (int i = 65;  i >= 11;  i--)   data[i-1] = data[i];

d)  for (int i = 65;  i >= 11;  i--)   data[i]   = data[i-1];

e)  none of the above
```

Correct Answer: a - This has the effect of deleting the object that had been at **data[10]**.

12. An **int** array named **data** holds live data, sorted into *ascending* order, in elements **0** through **n-1**, inclusive (i.e. there are **n** instances of live data, the first of which is stored in **data[0]** and the last of which is stored in **data[n-1]**). Which **for** loop will invert the data so that it remains in the same block of cells but in *descending* order.

ANSWERS:

a)
```
for (int i = 0;  i < n;  i++)
   {
      int holder = data[i];
      data[i]    = data[n-1-i];
      data[n-1-i]  = holder;
   }
```

b)
```
for (int i = 0;  i < n;  i++)
   {
      int holder = data[i];
      data[i]    = data[n-1];
      data[n-1]  = holder;
   }
```

c)
```
for (int i = 0;  i < n/2;  i++)
   {
      int holder = data[i];
      data[i]    = data[n-1-i];
      data[n-1-i]  = holder;
   }
```

d)
```
for (int i = 0;  i < n/2;  i++)
   {
      int holder = data[i];
      data[i]    = data[n-1];
      data[n-1]  = holder;
   }
```

Correct Answer: c - It is always important to visualize coded action. Here, the data will be inverted when "the lower half is swapped with upper half."

Once the **for** loop has gone through "the lower half" of the array, processing through the upper half un-inverts the data. It "swaps back" the values to their original compartments.

13. Which is the correct programming of a **static** method that accepts an **int** array as an argument and returns a **boolean** indicating whether or not the array's values are in nondescending (i.e. sorted) order?

ANSWERS:

a)
```
public static boolean isSorted()
   {
      boolean sorted = true;
      int i = 0;
      while(sorted  &&  index < a.length - 1)
        {
           if (this.a[i] > this.a[i+1])  sorted = false;
           i++;
        }
      return sorted;
   }
```

b)
```
public static boolean isSorted(int[] a)
   {
      boolean sorted = true;
      int i = 0;
      while(sorted  ||  index < a.length - 1)
        {
           if (a[i] > a[i+1])  sorted = false;
           i++;
        }
      return sorted;
   }
```

c)
```
public static boolean isSorted(int[] a)
   {
      boolean sorted = false;
      int i = 0;
      while(sorted  &&  index < a.length - 1)
        {
           if (a[i] <= a[i+1])  sorted = true;
           i++;
        }
      return sorted;
   }
```

d)
```
public static boolean isSorted(int[] a)
   {
      boolean sorted = true;
      int i = 0;
      while(sorted  &&  index < a.length - 1)
        {
           if (a[i] > a[i+1])  sorted = false;
           i++;
        }
      return sorted;
    }
```

Correct Answer: d

14. Objects within an array can be "logically sorted" instead of being physically moved around by using an auxiliary "tag array."

The tag array is a separate array, the same length as the data array. It stores subscripts for accessing the data array such that:

- **tag[0]** gives the subscript for the first item of data (e.g. the datum with the lowest value)
- **tag[1]** gives the subscript for the second item of data

and so on up through

- **tag[data.length - 1]** which gives the subscript of the nth (e.g. "the largest") item of data.

The diagram below illustrates the relationship between a data array and its tag array:

data :	34	18	31	26	24
	0	1	2	3	4

tag :	1	4	3	2	0
	0	1	2	3	4

Which segment of code captures the logic for displaying the data in sorted order?

ANSWERS:

a)
```
for (int i = 0;  i < tag.length;  i++)
   System.out.println( tag[ data[i] ] );
```

b)
```
for (int i = 0;  i < tag.length;  i++)
   System.out.println( data[ tag[i] ] );
```

c)
```
for (int i = 0;  i < tag.length;  i++)
   System.out.println( tag[i] );
```

d)
```
for (int i = 0;  i < tag.length;  i++)
   System.out.println( tag[i] );
```

Correct Answer: b - Tag sorts enhance performance when the objects are very large (or on spinning storage). Moving huge numbers of bits from one place to another is time-intensive. When you can avoid this you can improve performance.

15. The following code puts 1000 random integers within the range of 0..999 inclusive into an array:

```
int[] a = new int[1000];

for (int i = 0;  i < a.length;  i++)
  {
     a[i] = (int) (Math.random() * 1000);
  }
```

How would the well-versed Java programmer be inclined to sort it?

ANSWERS:

a) **`java.util.Arrays.sort( a );`**

b) **`java.util.Arrays.sort( a[] );`**

c) **`a.sort();     //presumes that java.util.Arrays was imported`**

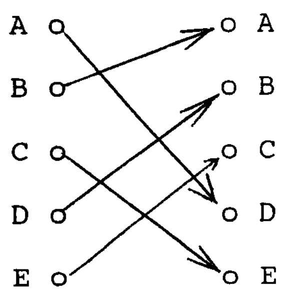
d) **`a[].sort();  //presumes that java.util.Arrays was imported`**

e) hire a consultant

Correct Answer: a - Java supplies a sorting method that is overloaded to handle arrays of **chars**, **shorts**, **bytes**, **longs**, **floats**, **doubles**, **Objects** that implement the **Comparable** interface, and **Objects** that do not if a **Comparator** is passed as an additional argument (i.e. an **Object** that implements the **Comparator** interface).

sort() is a **static** method in the **Arrays** class in the **Java.util** package.

16. Code is needed that performs the mapping illustrated below:



Values of 'A' are changed to 'D'

Values of 'B' are changed to 'A'

Values of 'C' are changed to 'E'

Values of 'D' are changed to 'B'

Values of 'E' are changed to 'C'

Which fragment(s) of code perform properly?

ANSWERS:

a)
```
if (value == 'A')  value = 'D';
if (value == 'B')  value = 'A';
if (value == 'C')  value = 'E';
if (value == 'D')  value = 'B';
if (value == 'E')  value = 'C';
```

b)
```
if (value == 'A' || value == 'a')  value = 'D';
if (value == 'B' || value == 'b')  value = 'A';
if (value == 'C' || value == 'c')  value = 'E';
if (value == 'D' || value == 'd')  value = 'B';
if (value == 'E' || value == 'e')  value = 'C';
```

c)
```
if (value == 'A')  value = 'D';
else  if (value == 'B')  value = 'A';
else  if (value == 'C')  value = 'E';
else  if (value == 'D')  value = 'B';
else  if (value == 'E')  value = 'C';
```

d)
```
if (value == 'A')  value = 'D';
  {
    if (value == 'B')  value = 'A';
      {
        if (value == 'C')  value = 'E';
          {
            if (value == 'D')  value = 'B';
              {
                 if (value == 'E')  value = 'C';
              }
          }
      }
  }
```

e) the fragments of code in both c and d are correct

Correct Answer: c

17. Given the following class:

```
public class Rectangle
  {
    public int length = 10;
    public int width  =  5;

    public Rectangle()
      {
      }

    public Rectangle(int length, int width)
      {
        this.length = length;
        this.width  = width;
      }

    public int area()
      {
        return length*width;
      }

    public String toString()
      {
         return "length = " + length + ", width = " + width";
      }

    //details omitted
  }
```

Java provides a **Stack** class in its **java.util** package with the following (abbreviated) application programming interface:

```
public class Stack extends Vector
  {
     //public constructor
      public Stack()

     //public instance methods
     public boolean empty()
     public Object pop()
     public Object push(Object item)
  }
```

17.a. Which of the following correctly creates five **Rectangle** objects and pushes them onto a **Stack**?

ANSWERS:

a)
```
java.util.Stack  stack =  new java.util.Stack();

for (int i = 0;  i < 5;  i++)
  {
     stack.push( new Rectangle() );
  }
```

b)
```
java.util.Stack  stack =  new java.util.Stack();

for (int i = 0;  i < 5;  i++)
  {
     stack.push( new Rectangle(2*(i+1), i+1) );
  }
```

c)
```
java.util.Stack  stack =  new java.util.Stack();

for (int i = 0;  i < 5;  i++)
  {
     Rectangle r = new Rectangle(2*(i+1), i+1);
     stack.push( r );
  }
```

d)
```
java.util.Stack  stack =  new java.util.Stack();

for (int i = 0;  i < 5;  i++)
  {
     Rectangle r = new Rectangle(2*(i+1), i+1);
     Object  obj = (Object) r;
     stack.push( obj );
  }
```

e)
```
java.util.Stack  stack =  new java.util.Stack();

for (int i = 0;  i < 5;  i++)
  {
     Rectangle r = new Rectangle();
     Object  obj = (Object) r;
     stack.push( obj )
  }
```

f) none of the above (nasty!)

Correct Answer: a, b, c, d, and e

17.b. Which of the following takes the top **Rectangle** off the **Stack** and displays the values of its **length** and **width** instance variables?

ANSWERS:

a)
```
if (! stack.empty())
  {
    Rectangle r = stack.pop();
    System.out.print( r.length + ", " + r.width);
  }
```

b)
```
if (! stack.empty())
  {
    Rectangle r = stack.pop();
    System.out.println( r.toString() );
  }
```

c)
```
if (! stack.empty())
  {
    Object obj = stack.pop();
    System.out.println( obj.length + ", " + obj.width);
  }
```

d)
```
if (! stack.empty())
  {
    Object obj = stack.pop();
    System.out.println( obj.toString() );
  }
```

e)
```
if (! stack.empty())
  {
    Object obj = stack.pop();
    System.out.println( obj );
  }
```

f)
```
if (! stack.empty())
  {
    Object obj    = stack.pop();
    Rectangle r   = (Rectangle) obj;
    System.out.println( r );
  }
```

g)
```
if (! stack.empty())
  {
    Object obj = stack.pop();
    Rectangle r = (Rectangle) obj;
    System.out.println( r.length + ", " + r.width);
  }
```

Correct Answer: d, e, f, and g

18. What is the output from the following program?

```
class Main18
 {
   public static void main(String[] args)
    {
      int[] a = {11, 12, 13, 14, 15};

      try
       {
         System.out.println( a[4] );
         System.out.println( a[5] );
       }
      catch(Exception e)
       {
         System.out.print("Error: last element is ");
         System.out.print("a[" + (a.length - 1) + "]");
         System.out.println();
       }

      System.out.println("Okay, let's go on.");
    }
 }
```

ANSWERS:

a) compilation fails because there is no element **a[5]**

b)
```
15
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException
at Main18.main(Main18.java:10)
```

c)
```
15
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException
at Main18.main(Main18.java:10)
Error:  last element is a[4]
```

d)
```
15
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException
at Main18.main(Main18.java:10)
Error: last element is a[4]
Okay, let's go on.
```

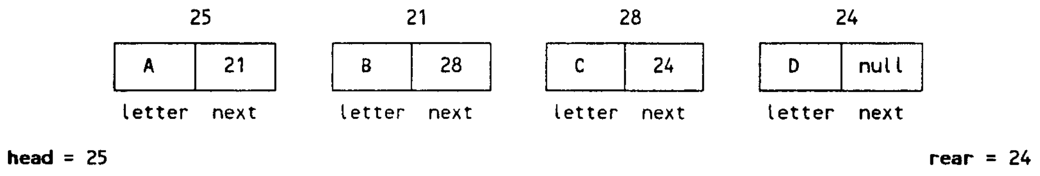
e) none of the above

Correct Answer: e - The output generated is:

```
15
Error: last element is a[4]
Okay, let's go on.
```

19. The depicted linked list is constructed from **Node** objects.

```
public class Node
  {
    public char letter = ' ';
    public Node next   = null;
  }
```



The numbers depicted above the nodes are metaphorical addresses. What could we code to see the actual memory locations the external **head** and **rear** references as well as of each node on the list? (Naturally, the code must be non-destructive.)

ANSWERS:

a)
```
System.out.println("external head:  " + head);
System.out.println();
System.out.println("the list:)
while ( head != null )
  {
     System.out.println(head);
     head = head.next;
  }
System.out.println();
System.out.println("external rear:  " + rear);
```

b)
```
System.out.println("external head:  " + head);
System.out.println();
System.out.println("the list:)
Node cursor = head;
while ( cursor != null )
  {
     System.out.println(cursor);
     cursor = cursor.next;
  }
System.out.println();
System.out.println("external rear:  " + rear);
```

c)
```
System.out.println("external head:  " + head);
System.out.println();
System.out.println("the list:)
Node cursor = head;
while ( cursor.next != rear )
  {
     cursor = cursor.next;
     System.out.println(cursor);
  }
System.out.println();
System.out.println("external rear:  " + rear);
```

d) Trick question! Unlike C and C++, Java is a "pointerless" langauge. It will not display addresses.

Correct Answer: b - a seems the same as b, but it leaves head storing null, not the address of the list's first node

Address are displayed like this: Node@1036ef54

20. The following code would display the letters in the previous item's linked list from front to back:

```
Node cursor = head;
while ( cursor != null )
  {
    System.out.println(cursor.letter);
    cursor = cursor.next;
  }
```

How could the letters in the list be displayed from back to front?

ANSWERS:

a) they can't -- this is a singly linked list, not a doubly linked list

b) they can, but unfortunately it is always an $O(n^2)$ operation:

```
Node lastProcessed = null;
while (lastProcessed != head)
  {
    //move cursor to node before lastProcessed
    Node cursor = head;
    while (cursor.next != lastProcessed)
      {
        cursor = cursor.next;
      }

    System.out.print( cursor.letter );
    lastProcessed = cursor;
  }
System.out.print( cursor.letter );
```

c) they can, but it requires the explicit use of a **Stack**:

```
java.util.Stack stack = new java.util.Stack();

//load the stack
Node cursor = head;
while (cursor != null)
  {
    stack.push(cursor);
    cursor = cursor.next;
  }

//process from the stack
while ( !stack.empty() )
  {
    cursor = (Node) (stack.pop());
    System.out.print(cursor.letter);
  }
```

d) they can, but it requires recursion:

```
void backwards(Node cursor)
  {
    if (cursor.next != null)  backwards(cursor.next);
    System.out.print( cursor.letter );
  }
```

e) the code in b, c, and d is correct in each case
but the prose are fallacious

f) none of the above

Correct Answer: e

Computer Trivia

These questions were addressed to everyone present. A "fun-sized" chocolate bar was thrown to the first person who shouted out a correct answer. Their order here differs from the order in which they were presented.

** What was the first programming language with objects, and and around when did it appear?

Answer: Simula, 1967

** Back in the days when PL/1 and Pascal were widely used in algorithms and compiler classes, radical professors would fail a program if it used a certain, perfectly legal statement. Which statement was this?

Answer: the GOTO statement

** What is a seasoned programmer's term an infinite loop?

Answer: A dynamic halt

** Complete the following truism about building programs: **"The sooner you start to code, ..."**

Answer: **the longer the program will take** -- Proclaimed by Roy Carlson, at the University of Wisconsin, and cited by Jon Bentley in <u>More Programming Pearls</u>: <u>Confessions of a Coder</u>, Addison-Wesley, 1988, ISBN: 0-201-11889-0, page 58.

** Complete the following truism about debugging: **"Testing can show the presence of bugs, ..."**

Answer: **but not their absence.** -- Proclaimed by Edsger W. Dijkstra; cited by Bentley in the reference above on page 60.

** Complete the following truism about enhancing a program's performance: **"Don't diddle code to make it faster, ..."**

Answer: **find a better algorithm.** -- Proclaimed by Brian W. Kernighan and P.J. Plauger in <u>The Elements of Programming Style</u> (Second Edition), McGraw-Hill, 1978, ISBN: 0-07-034207-5, page 161.

** Who was "the father" of the mouse (or, should we phrase this as, its inventor)?

Answer: Doug Engelbart and, in his patent, he called it the "x-y position indicator."

** Modifications in requirement specifications are the bane of the software developer. Contemporary developers acknowledge that **change is inevitable**, but with one **exception**, which is what?

Answer: **from vending machines** Roger S. Pressman, in <u>Software Engineering: A Practitioner's Approach</u> (Fifth Edition), McGraw-Hill, 2001, ISBN: 0-07-365578-3, page 236, informs us that "Change is inevitable, except from vending machines."

** Edsger W. Dijkstra said the following about which widely-used programming language?

"The use of [blank] cripples the mind; its teaching should, therefore, be regarded as a criminal offence."

Answer: COBOL Essay EWD498, "How Do We Tell Truths that Might Hurt?" in <u>Selected Writings on Computing: A Personal Perspective</u>, Springer-Verlag (1982), ISBN: 0-387-90652-5, page 130.

** Who wrote the book, Algorithms + Data Structures = Programs, and how did the author spell "plus" and "equals" in its title?

Answer: Niklaus Wirth, he used the plus sign and the equals sign.

** What programming language was touted by its developer, when introduced, as "a better C"?

Answer: C++

** Give the last name of one of the "gang of four" and state what they did.

Answer: The gang of four is Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides.

They wrote a book titled <u>Design Patterns: Elements of Reusable Object-Oriented Software</u> Addison Wesley Longman (1995), ISBN: 0-201-63361-2.

A design pattern, or pattern for short, is described by Peter van der Linden as "a set of steps for doing something, like a recipe is a set of steps for cooking something." He goes on, "...design patterns describe simple, repeatable solutions to specific problems in object-oriented software design. They capture solutions that have been improved over time; hence they aren't typically the first code that comes to mind.... They are code idioms write large. They are not unusual or amazing, or tied to any one langauge." [<u>Just Java 2</u> (Fifth Edition), Prentice Hall (2002), ISBN: 0-13-032072-2, page 307]

** The old "ten-ninety" rule states that in a prototype shakedown, 10% of the bugs take up 90% of the time. What is Frederick Brooks' rule about the impact of adding programmers to a project to offset schedule slippage?

Answer: Adding programmers to a project that is late makes it later. [Frederick P. Brooks, Jr., <u>The Mythical Man-Month</u> (Anniversary Edition), Addison-Wesley (1995), ISBN: 0-201-83595-9, page 14 and also page 232]

** What does API stand for?

Answer: Application Programming Interface

** What does IDE stand for?

Answer: Integrated Development Environment

** Which one individual was chiefly responsible for Java?

Answer: James Gosling

** Which one individual was chiefly responsible for COBOL?

Answer: the naval commodore, Grace Murray Hopper

** How far will an electrical signal travel in a nanosecond, which is one one-billionth of a second?

Answer: around a foot, which is the same as light

** Which highly esteemed computer scientist has written books and lectured on the Bible and religion?

Answer: Donald E. Knuth One book is _3:16 Bible Texts Illuminated_, another is _Things a Computer Scientist Rarely Talks About_

** Donald Knuth, whose volumes on the _Art of Computer Programming_ are credited with founding the study of data structures and algorithms, had his first publication in which periodical?

Answer: _Mad Magazine_

** Which one individual was chiefly responsible for C?

Answer: Dennis Ritchie

** Which one individual was chiefly responsible for C++?

Answer: Bjarne Stroustrup

** Which one individual was chiefly responsible for Pascal?

Answer: Niklaus Wirth

** What is Forte?

Answer: a Java IDE available for free from the Sun Microsystems website, java.sun.com

** What does J2ME stand for?

Answer: Java 2 Micro Edition

** What does J2SE stand for?

Answer: Java 2 Standard Edition

** Java 1.4 was released in December 2001. When is Java 1.5 slated for release?

Answer: July 2003

** What organization controls the evolution of the Java platform (i.e. the content the language)?

Answer: Not Sun Microsystems; rather,
The Java Community Process (i.e. the JCP),
which is an open forum of around 300 companies.
IBM plays a big part (but does not control it).
Anyone can join the JCP.

** The Java foundation classes relate to what sphere of programming?

Answer: graphics and graphical user interfaces

** The java enterprise classes relate to what sphere of programming?

Answer: distributed computing (groups of programs interacting over a network)

** What sorting algorithm was used for electronic data processing prior to 1930 and had, as its time complexity, a big-oh of N?

Answer: the radix sort (used to sort punch cards on mechanical sorters)

** The RGB color scheme is used to describe colors with respect to the emission of light from subpixels. What does RGB stand for?

Answer: red, green, blue

** In the RGB color system, give the three numbers in base 10 that represents the yellowest yellow.

Answer: 255, 255, 0

** Java's mascot looks like a tooth with a big round red nose. What's his name?

Answer: Duke

** What does CORBA stand for?

Answer: Common Object Request Broker Architecture

** In today's world of distributed objects, what does SOAP stand for?

Answer: Simple Object Access Protocol

** It would seem that computing is running out of acronyms. In contemporary computing, SOAP stands for Simple Object Access Protocol. What did SOAP stand for long about 1956 when the dominant computer in commercial use was the IBM 650?

Answer: an assembly language for the IBM 650, Symbolic Optimum Assembly Programming

** Approximately what percentage of software development projects fail (i.e. collapsed before delivering a product)?

Answer: 26% (and 46% of those that do deliver a product experience cost and schedule overruns) [Roger S. Pressman, <u>Software Engineering: A Practitioner's Approach</u> (Fifth Edition), McGraw-Hill, 2001, ISBN: 0-07-365578-3, page 57]

** With the Java SDK installed, what command at the DOS prompt, followed by a fully qualified class's name, will display the class's API?

Answer: javap For example, the following:
A:\>javap java.util.StringTokenizer

** Smalltalk, a pure object-oriented programming language that sold itself with the slogan "programming by extension rather than reinvention, predated C++ by fully ten years. Who developed it?

Answer: three researchers at the Xerox Palo Alto Research Center (PARC): Alan Kay, Adele Goldberg, and Dan Ingalls

** The character with the ASCII value of 47 is sometimes referred to as the virgule. What would a normal person call it?

Answer: the slash, diagonal, solidus (as in "yes/no") or the division operator (as in "sum/n")

** The character with the ASCII value of 35 is sometimes referred to as the octothorp. What would a normal person call it?

Answer: the pound sign or the number sign (#)

** Give the Web site of an online encyclopedia of computer technology.

Answer: http://webopedia.com

** Andy Bechtolsheim, Vinod Khosla, and Scott McNealy; graduate students at Stanford University; along with Bill Joy from Berkeley started a company. It was based on selling a workstation built on the Motorola 68000 processor that worked on the Stanford University Network. What was the name of their workstation and their company?

Answer: the S.U.N. workstation (for Stanford University Network); the company is Sun Microsystems

** The original IBM Personal Computer was based on the Intel 8088 microprocessor. What was its speed?

Answer: 4.77 megahertz

** Moore's law, states that the number of transistors that can be built on the same sized piece of silicon will double every X months. Reporters often re-phrase this, saying that the processing power of a chip doubles every X months. What number is X?

Answer: 18

** MP3 (or, more formally, "MPEG-1, audio layer 3") refers to a compressed format for storing and transmitting music. Approximately how much larger is a CD-quality file than an MP3 file storing the same cut?

Answer: 12 times larger

** Who is credited with inventing and developing the relational database?

Answer: E. F. Codd

** The following is a three-part question pertaining to the world's first "computer lab." Just as DoIT calls our labs "Computer Resource Centers," this lab was called the "Digital Computer Center."

At what University was it founded?

Answer: Purdue University

What year was it founded?

Answer: 1952

Who was responsible?

Answer: Alan J. Perlis

** Who is often credited as being the world's first commercial programmer (that is, the first individual to earn his salary by writing programs for a digital computer)?

Answer: Edsger W. Dijkstra
He started work in the Spring of 1952 in the Netherlands. He tells this story: "...in 1957 I married, and Dutch marriage rites require you to state your profession and I stated that I was a programmer. But the municipal authorities of the town of Amsterdam did not accept it on the grounds that there was no such profession."
[for Dijkstra's 1972 Turing Award Lecture]

** The 3½ inch diskette is officially obsolete. A floppy drive is no longer standard equipment on new computers. How many hundreds of million 3 1/2" diskettes were sold in the United States last year?

Answer: between 400 and 500 million diskettes

** People who want faster Internet connectivity at home are debating between cable and DLS. What does DSL stand for?

Answer: digital subscriber line

** The radix sort was the first sorting algorithm to see wide commercial use. Why was this?

Answer: Because it was quiquely suited to the mechanical sorting machines, which could drop cards into binds based on the punch in a selected column. Knuth reports that the radix sort was used in data processing throughout the United States in the 1920s.

** What was the first sorting algorithm programmed for a true, stored-program digital computer?

Answer: a mergesort, and it was programmed by John Von Neumann in 1945