Diagnostic Test

COMPUTER SCIENCE A
SECTION I

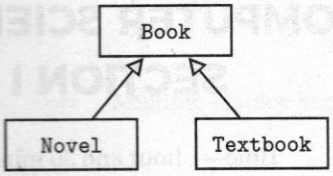
Time—1 hour and 30 minutes
Number of questions—40
Percent of total grade—50

DIRECTIONS: Determine the answer to each of the following questions or incomplete statements, using the available space for any necessary scratchwork. Then decide which is the best of the choices given and fill in the corresponding oval on the answer sheet. Do not spend too much time on any one problem.

NOTES:

- Assume that the classes in the Quick Reference have been imported where needed.
- Assume that variables and methods are declared within the context of an enclosing class.
- Assume that method calls that have no object or class name prefixed, and that are not shown within a complete class definition, appear within the context of an enclosing class.
- Assume that parameters in method calls are not `null` unless otherwise stated.

1. Consider this inheritance hierarchy, in which `Novel` and `Textbook` are subclasses of `Book`.



Which of the following is a false statement about the classes shown?

(A) The `Textbook` class can have private instance variables that are in neither `Book` nor `Novel`.

(B) Each of the classes—`Book`, `Novel`, and `Textbook`—can have a method `computeShelfLife`, whose code in `Book` and `Novel` is identical, but different from the code in `Textbook`.

(C) If the `Book` class has private instance variables `title` and `author`, then `Novel` and `Textbook` cannot directly access them.

(D) Both `Novel` and `Textbook` inherit the constructors in `Book`.

(E) If the `Book` class has a private method called `readFile`, this method may not be accessed in either the `Novel` or `Textbook` classes.

GO ON TO THE NEXT PAGE.

2. A programmer is designing a program to catalog all books in a library. She plans to have a `Book` class that stores features of each book: `author`, `title`, `isOnShelf`, and so on, with operations like `getAuthor`, `getTitle`, `getShelfInfo`, and `setShelfInfo`. Another class, `LibraryList`, will store an array of `Book` objects. The `LibraryList` class will include operations such as `listAllBooks`, `addBook`, `removeBook`, and `searchForBook`. What is the relationship between the `LibraryList` and `Book` classes?
   (A) Composition
   (B) Inheritance
   (C) Independent classes
   (D) Polymorphism
   (E) `ArrayList`

3. Consider the following code segment, which is intended to add zero to the end of `list` every time a certain condition is met. You may assume that `list` is an `ArrayList<Integer>` that contains at least one element.

```
for (Integer num : list)
{
    if (<condition>)
        list.add(0);
}
```

   Which of the following errors is most likely to occur?
   (A) `ArrayIndexOutOfBoundsException`
   (B) `IndexOutOfBoundsException`
   (C) `NullPointerException`
   (D) `ConcurrentModificationException`
   (E) `IllegalArgumentException`

GO ON TO THE NEXT PAGE.

Questions 4 and 5 refer to the `Card` and `Deck` classes shown below.

```
public class Card
{
    private String suit;
    private int value;     //0 to 12

    public Card(String cardSuit, int cardValue)
    { /* implementation */ }

    public String getSuit()
    { return suit; }

    public int getValue()
    { return value; }

    public String toString()
    {
        String faceValue = "";
        if (value == 11)
            faceValue = "J";
        else if (value == 12)
            faceValue = "Q";
        else if (value == 0)
            faceValue = "K";
        else if (value == 1)
            faceValue = "A";
        if (value >= 2 && value <= 10)
            return value + " of " + suit;
        else
            return faceValue + " of " + suit;
    }
}

public class Deck
{
    private Card[] deck;
    public final static int NUMCARDS = 52;

    public Deck()
    { ...

    /** Simulate shuffling the deck. */
    public void shuffle()
    { ...

    //Other methods are not shown.
}
```

GO ON TO THE NEXT PAGE.

4. Which of the following represents correct /* *implementation* */ code for the constructor in the Card class?

(A)
```
suit = cardSuit;
value = cardValue;
```

(B)
```
cardSuit = suit;
cardValue = value;
```

(C)
```
Card = new Card(suit, value);
```

(D)
```
Card = new Card(cardSuit, cardValue);
```

(E)
```
suit = getSuit();
value = getValue();
```

5. Consider the implementation of a writeDeck method that is added to the Deck class.

```
/** Write the cards in deck, one per line. */
public void writeDeck()
{
    /* implementation code */
}
```

Which of the following is correct /* *implementation code* */?

I
```
System.out.println(deck);
```

II
```
for (Card card : deck)
    System.out.println(card);
```

III
```
for (Card card : deck)
    System.out.println((String) card);
```

(A) I only
(B) II only
(C) III only
(D) I and III only
(E) II and III only

GO ON TO THE NEXT PAGE.

6. Refer to the following method that finds the smallest value in an array.

```
/** Precondition: arr is an array of nonzero length
 *                and is initialized with int values.
 *  @param arr the array to be processed
 *  @return the smallest value in arr
 */
public static int findMin(int[] arr)
{
    int min = /* some value */;
    int index = 0;
    while (index < arr.length)
    {
        if (arr[index] < min)
            min = arr[index];
        index++;
    }
    return min;
}
```

Which replacement(s) for `/* some value */` will always result in correct execution of the `findMin` method?

I `Integer.MIN_VALUE`

II `Integer.MAX_VALUE`

III `arr[0]`

(A) I only
(B) II only
(C) III only
(D) I and III only
(E) II and III only

GO ON TO THE NEXT PAGE.

7. Consider the following loop, where `n` is some positive integer.

```
for (int i = 0; i < n; i += 2)
{
    if (/* test */)
        /* perform some action */
}
```

In terms of `n`, which Java expression represents the maximum number of times that `/* perform some action */` could be executed?

(A) `n / 2`
(B) `(n + 1) / 2`
(C) `n`
(D) `n - 1`
(E) `(n - 1) / 2`

8. A method is to be written to search an array for a value that is larger than a given item and return its index. The problem specification does not indicate what should be returned if there are several such values in the array. Which of the following actions would be best?

(A) The method should be written on the assumption that there is only one value in the array that is larger than the given item.
(B) The method should be written so as to return the index of every occurrence of a larger value.
(C) The specification should be modified to indicate what should be done if there is more than one index of larger values.
(D) The method should be written to output a message if more than one larger value is found.
(E) The method should be written to delete all subsequent larger items after a suitable index is returned.

GO ON TO THE NEXT PAGE.

9. When will method `whatIsIt` cause a stack overflow (i.e., cause computer memory to be exhausted)?

```
public static int whatIsIt(int x, int y)
{
    if (x > y)
        return x * y;
    else
        return whatIsIt(x - 1, y);
}
```

(A) Only when $x < y$
(B) Only when $x \leq y$
(C) Only when $x > y$
(D) For all values of `x` and `y`
(E) The method will never cause a stack overflow.

10. The boolean expression `a[i] == max || !(max != a[i])` can be simplified to
(A) `a[i] == max`
(B) `a[i] != max`
(C) `a[i] < max || a[i] > max`
(D) `true`
(E) `false`

11. Consider the following code segment.

```
int[][] mat = {{3,4,5},
               {6,7,8}};

int sum = 0;
for (int[] arr: mat)
{
    for (int n = 0; n < mat.length; n++)
        sum += arr[n];
}
```

What is the value of `sum` as a result of executing the code segment?
(A) 9
(B) 11
(C) 13
(D) 20
(E) 33

GO ON TO THE NEXT PAGE.

12. Consider a `Clown` class that has a default constructor. Suppose a list `ArrayList<Clown> list` is initialized. Which of the following will not cause an `IndexOutOfBoundsException` to be thrown?

(A)
```
for (int i = 0; i <= list.size(); i++)
    list.set(i, new Clown());
```

(B)
```
list.add(list.size(), new Clown());
```

(C)
```
Clown c = list.get(list.size());
```

(D)
```
Clown c = list.remove(list.size());
```

(E)
```
list.add(-1, new Clown());
```


GO ON TO THE NEXT PAGE.

Refer to the following class for Questions 13 and 14.

```
public class Tester
{
    private int[] testArray = {3, 4, 5};

    /** @param n an int to be incremented by 1 */
    public void increment (int n)
    { n++; }

    public void firstTestMethod()
    {
        for (int i = 0; i < testArray.length; i++)
        {
            increment(testArray[i]);
            System.out.print(testArray[i] + " ");
        }
    }

    public void secondTestMethod()
    {
        for (int element : testArray)
        {
            increment(element);
            System.out.print(element + " ");
        }
    }
}
```

13. What output will be produced by invoking `firstTestMethod` for a `Tester` object?

    (A) `3 4 5`
    (B) `4 5 6`
    (C) `5 6 7`
    (D) `0 0 0`
    (E) No output will be produced. An `ArrayIndexOutOfBoundsException` will be thrown.

14. What output will be produced by invoking `secondTestMethod` for a `Tester` object, assuming that `testArray` contains `3,4,5`?

    (A) `3 4 5`
    (B) `4 5 6`
    (C) `5 6 7`
    (D) `0 0 0`
    (E) No output will be produced. An `ArrayIndexOutOfBoundsException` will be thrown.

GO ON TO THE NEXT PAGE.

Questions 15–17 refer to the following `Point`, `Quadrilateral`, and `Rectangle` classes.

```
public class Point
{
    private int xCoord;
    private int yCoord;

    //constructor
    public Point(int x, int y)
    {
        ...
    }

    //accessors

    public int get_x()
    {
        ...
    }

    public int get_y()
    {
        ...
    }

    //Other methods are not shown.

}

public class Quadrilateral
{
    private String labels;    //e.g., "ABCD"

    //constructor
    public Quadrilateral(String quadLabels)
    { labels = quadLabels; }

    public String getLabels()
    { return labels; }

    public int perimeter()
    { return 0; }

    public int area()
    { return 0; }
}
```


GO ON TO THE NEXT PAGE.

```
public class Rectangle extends Quadrilateral
{
    private Point topLeft;   //coords of top left corner
    private Point botRight;  //coords of bottom right corner

    //constructor
    public Rectangle(String theLabels, Point theTopLeft, Point theBotRight)
    { /* implementation code */ }

    public int perimeter()
    { /* implementation not shown */ }

    public int area()
    { /* implementation not shown */ }

    //Other methods are not shown.
}
```

15. Which of the following statements about the `Point`, `Quadrilateral`, and `Rectangle` classes are false?

    I `Point` is a subclass of `Quadrilateral`.

    II `Point` is a subclass of `Rectangle`.

    III The `Rectangle` class inherits the constructor of `Quadrilateral`.

    (A) I only
    (B) II only
    (C) III only
    (D) I and II only
    (E) I, II, and III

16. Which represents correct `/* implementation code */` for the `Rectangle` constructor?

    I
    ```
    super(theLabels);
    ```

    II
    ```
    super(theLabels, theTopLeft, theBotRight);
    ```

    III
    ```
    super(theLabels);
    topLeft = theTopLeft;
    botRight = theBotRight;
    ```

    (A) I only
    (B) II only
    (C) III only
    (D) I and II only
    (E) II and III only

GO ON TO THE NEXT PAGE.

17. Refer to the `Parallelogram` and `Square` classes below.

```
public class Parallelogram extends Quadrilateral
{
    //Private instance variables and constructor are not shown.
        ...

    public int perimeter()
    { /* implementation not shown */ }

    public int area()
    { /* implementation not shown */ }
}

public class Square extends Rectangle
{
    //Private instance variables and constructor are not shown.
        ...

    public int perimeter()
    { /* implementation not shown */ }

    public int area()
    { /* implementation not shown */ }
}
```

Consider an `ArrayList<Quadrilateral> quadList` whose elements are of type `Rectangle`, `Parallelogram`, or `Square`.
Refer to the following method, `writeAreas`:

```
/** Precondition: quadList contains Rectangle, Parallelogram, or
 *                Square objects in an unspecified order.
 */
public static void writeAreas(ArrayList<Quadrilateral> quadList)
{
    for (Quadrilateral quad : quadList)
        System.out.println("Area of " + quad.getLabels()
            + " is " + quad.area());
}
```

What is the effect of executing this method?

(A) The area of each `Quadrilateral` in `quadList` will be printed.
(B) A value of `0` will be printed for each element of `quadList`.
(C) A compile-time error will occur, stating that there is no `getLabels` method in classes `Rectangle`, `Parallelogram`, or `Square`.
(D) A `NullPointerException` will be thrown.
(E) A `ConcurrentModificationException` will occur.

GO ON TO THE NEXT PAGE.

18. Refer to the `doSomething` method below.

```
// postcondition
public static void doSomething(ArrayList<SomeType> list, int i, int j)
{
    SomeType temp = list.get(i);
    list.set(i, list.get(j));
    list.set(j, temp);
}
```

Which best describes the ***postcondition*** for `doSomething`?

(A) Removes from `list` the objects indexed at `i` and `j`.

(B) Replaces in `list` the object indexed at `i` with the object indexed at `j`.

(C) Replaces in `list` the object indexed at `j` with the object indexed at `i`.

(D) Replaces in `list` the objects indexed at `i` and `j` with `temp`.

(E) Interchanges in `list` the objects indexed at `i` and `j`.

GO ON TO THE NEXT PAGE.

19. Consider the `NegativeReal` class below, which defines a negative real number object.

```
public class NegativeReal
{
    private Double negReal;

    /** Constructor. Creates a NegativeReal object whose value is num.
     *  @param num  a negative real number
     */
    public NegativeReal(double num)
    { /* implementation not shown */ }

    /** @return the value of this NegativeReal */
    public double getValue()
    { /* implementation not shown */ }

    /** @return this NegativeReal rounded to the nearest integer */
    public int getRounded()
    { /* implementation */ }
}
```

Here are some rounding examples:

Negative real number	Rounded to nearest integer
−3.5	−4
−8.97	−9
−5.0	−5
−2.487	−2
−0.2	0

Which `/* implementation */` of `getRounded` produces the desired postcondition?

(A) `return (int) (getValue() - 0.5);`

(B) `return (int) (getValue() + 0.5);`

(C) `return (int) getValue();`

(D) `return (double) (getValue() - 0.5);`

(E) `return (double) getValue();`

GO ON TO THE NEXT PAGE.

20. Consider the following method.

```
public static void whatsIt(int n)
{
    if (n > 10)
        whatsIt(n / 10);
    System.out.print(n % 10);
}
```

What will be output as a result of the method call `whatsIt(347)`?

(A) 74
(B) 47
(C) 734
(D) 743
(E) 347

21. A large list of numbers is to be sorted into ascending order. Assuming that a "data movement" is a swap or reassignment of an element, which of the following is a true statement?

(A) If the array is initially sorted in descending order, then insertion sort will be more efficient than selection sort.
(B) The number of comparisons for selection sort is independent of the initial arrangement of elements.
(C) The number of comparisons for insertion sort is independent of the initial arrangement of elements.
(D) The number of data movements in selection sort depends on the initial arrangement of elements.
(E) The number of data movements in insertion sort is independent of the initial arrangement of elements.

GO ON TO THE NEXT PAGE.

22. Refer to the definitions of `ClassOne` and `ClassTwo` below.

```
public class ClassOne
{
    public void methodOne()
    {
        ...
    }

    //Other methods are not shown.
}

public class ClassTwo extends ClassOne
{
    public void methodTwo()
    {
        ...
    }

    //Other methods are not shown.
}
```

Consider the following declarations in a client class. You may assume that `ClassOne` and `ClassTwo` have default constructors.

```
ClassOne c1 = new ClassOne();
ClassOne c2 = new ClassTwo();
```

Which of the following method calls will cause an error?

I `c1.methodTwo();`

II `c2.methodTwo();`

III `c2.methodOne();`

(A) None
(B) I only
(C) II only
(D) III only
(E) I and II only

GO ON TO THE NEXT PAGE.

23. Consider the code segment

```
if (n == 1)
    k++;
else if (n == 4)
    k += 4;
```

Suppose that the given segment is rewritten in the form

```
if (/* condition */)
    /* assignment statement */;
```

Given that `n` and `k` are integers and that the rewritten code performs the same task as the original code, which of the following could be used as (1) `/* condition */` and (2) `/* assignment statement */`?

(A) (1) `n == 1 && n == 4` (2) `k += n`

(B) (1) `n == 1 && n == 4` (2) `k += 4`

(C) (1) `n == 1 || n == 4` (2) `k += 4`

(D) (1) `n == 1 || n == 4` (2) `k += n`

(E) (1) `n == 1 || n == 4` (2) `k = n - k`

24. Which of the following will execute *without* throwing an exception?

I
```
String s = null;
String t = "";
if (s.equals(t))
    System.out.println("empty strings?");
```

II
```
String s = "holy";
String t = "moly";
if (s.equals(t))
    System.out.println("holy moly!");
```

III
```
String s = "holy";
String t = s.substring(4);
System.out.println(s + t);
```

(A) I only
(B) II only
(C) III only
(D) I and II only
(E) II and III only

GO ON TO THE NEXT PAGE.

25. Three numbers a, b, and c are said to be a *Pythagorean Triple* if and only if the sum of the squares of two of the numbers equals the square of the third. A programmer writes a method `isPythTriple` to test if its three parameters form a Pythagorean Triple:

```
//Returns true if a * a + b * b == c * c; otherwise returns false.
public static boolean isPythTriple(double a, double b, double c)
{
    double d = Math.sqrt(a * a + b * b);
    return d == c;
}
```

When the method was tested with known Pythagorean Triples, `isPythTriple` sometimes erroneously returned `false`. What was the most likely cause of the error?

(A) Round-off error was caused by calculations with floating-point numbers.
(B) Type `boolean` was not recognized by an obsolete version of Java.
(C) An overflow error was caused by entering numbers that were too large.
(D) `c` and `d` should have been cast to integers before testing for equality.
(E) Bad test data were selected.

26. Refer to the following class containing the `mystery` method.

```
public class SomeClass
{
    private int[] arr;

    /** Constructor. Initializes arr to contain nonnegative
     *  integers k such that 0 <= k <= 9.
     */
    public SomeClass()
    { /* implementation not shown */ }

    public int mystery()
    {
        int value = arr[0];
        for (int i = 1; i < arr.length; i++)
            value = value * 10 + arr[i];
        return value;
    }
}
```

Which best describes what the `mystery` method does?

(A) It sums the elements of `arr`.
(B) It sums the products `10*arr[0]` + `10*arr[1]` + ⋯ + `10*arr[arr.length-1]`.
(C) It builds an integer of the form $d_1 d_2 d_3 \ldots d_n$, where d_1 = `arr[0]`, d_2 = `arr[1]`, …, d_n = `arr[arr.length-1]`.
(D) It builds an integer of the form $d_1 d_2 d_3 \ldots d_n$, where d_1 = `arr[arr.length-1]`, d_2 = `arr[arr.length-2]`, …, d_n = `arr[0]`.
(E) It converts the elements of `arr` to base-10.

GO ON TO THE NEXT PAGE.

Questions 27 and 28 refer to the `search` method in the `Searcher` class below.

```
public class Searcher
{
    private int[] arr;

    /** Constructor. Initializes arr with integers. */
    public Searcher()
    { /* implementation not shown */ }

    /** Precondition: arr[first]...arr[last] sorted in ascending order.
     * Postcondition: Returns index of key in arr. If key not in arr,
     *                returns -1.
     */
    public int search(int first, int last, int key)
    {
        int mid;
        while (first <= last)
        {
            mid = (first + last) / 2;
            if (arr[mid] == key)          //found key, exit search
                return mid;
            else if (arr[mid] < key)      //key to right of arr[mid]
                first = mid + 1;
            else                          //key to left of arr[mid]
                last = mid - 1;
        }
        return -1;                        //key not in list
    }
}
```

27. Which assertion is true just before each execution of the `while` loop?
    (A) `arr[first]` < `key` < `arr[last]`
    (B) `arr[first]` ≤ `key` ≤ `arr[last]`
    (C) `arr[first]` < `key` < `arr[last]` or `key` is not in `arr`
    (D) `arr[first]` ≤ `key` ≤ `arr[last]` or `key` is not in `arr`
    (E) `key` ≤ `arr[first]` or `key` ≥ `arr[last]` or `key` is not in `arr`

28. Consider the array `a` with values

    4, 7, 19, 25, 36, 37, 50, 100, 101, 205, 220, 271, 306, 321

    where 4 is `a[0]` and 321 is `a[13]`. Suppose that the `search` method is called with `first` = 0 and `last` = 13 to locate the `key` 205. How many iterations of the `while` loop must be made in order to locate it?
    (A) 3
    (B) 4
    (C) 5
    (D) 10
    (E) 13

GO ON TO THE NEXT PAGE.

29. Consider the following `RandomList` class.

```
public class RandomList
{
    private int[] ranList;

    public RandomList()
    { ranList = getList(); }

    /** @return array with random Integers from 0 to 100
     *    inclusive */
    public int[] getList()
    {
        System.out.println("How many integers? ");
        int listLength = ...;      //read user input
        int[] list = new int[listLength];
        for (int i = 0; i < listLength; i++)
        {
            /* code to add integer to list */
        }
        return list;
    }

    /** Print all elements of this list. */
    public void printList()
    { ...
}
```

Which represents correct `/* code to add integer to list */`?

(A) `list[i] = (int) (Math.random() * 101);`

(B) `list.add((int) (Math.random() * 101));`

(C) `list[i] = (int) (Math.random() * 100);`

(D) `list.add(new Integer(Math.random() * 100))`

(E) `list[i] = (int) (Math.random() * 100) + 1;`

GO ON TO THE NEXT PAGE.

30. Refer to method `insert` described here. The `insert` method has two string parameters and one integer parameter. The method returns the string obtained by inserting the second string into the first starting at the position indicated by the integer parameter `pos`. For example, if `str1` contains `xy` and `str2` contains `cat`, then

```
insert(str1, str2, 0)   returns   catxy
insert(str1, str2, 1)   returns   xcaty
insert(str1, str2, 2)   returns   xycat
```

Method `insert` follows:

```
/** Precondition:  0 <= pos <= str1.length().
 *  Postcondition: If str1 = a0 a1 ... an-1 and str2 = b0 b1 ... bm-1,
 *                 returns a0 a1 ... apos-1 b0 b1 ... bm-1 apos apos+1 ... an-1
 */
public static String insert(String str1, String str2, int pos)
{
    String first, last;
       /* more code */
    return first + str2 + last;
}
```

Which of the following is a correct replacement for `/* more code */`?

(A)
```
first = str1.substring(0, pos);
last = str1.substring(pos);
```

(B)
```
first = str1.substring(0, pos - 1);
last = str1.substring(pos);
```

(C)
```
first = str1.substring(0, pos + 1);
last = str1.substring(pos + 1);
```

(D)
```
first = str1.substring(0, pos);
last = str1.substring(pos + 1, str1.length());
```

(E)
```
first = str1.substring(0, pos);
last = str1.substring(pos, str1.length() + 1);
```

GO ON TO THE NEXT PAGE.

31. A matrix (two-dimensional array) is declared as

```
int[][] mat = new int[2][3];
```

Consider the following method.

```
public static void changeMatrix(int[][] mat)
{
    for (int r = 0; r < mat.length; r++)
        for (int c = 0; c < mat[r].length; c++)
            if (r == c)
                mat[r][c] = Math.abs(mat[r][c]);
}
```

If `mat` is initialized to be

```
-1 -2 -6
-2 -4  5
```

which matrix will be the result of a call to `changeMatrix(mat)`?

(A)
```
 1 -2 -6
-2  4  5
```

(B)
```
-1  2 -6
 2 -4  5
```

(C)
```
-1 -2 -6
-2 -4 -5
```

(D)
```
 1 2 -6
 2 4  5
```

(E)
```
1 2 6
2 4 5
```


GO ON TO THE NEXT PAGE.

Use the following program description for Questions 32–34.

A programmer plans to write a program that simulates a small bingo game (no more than six players). Each player will have a bingo card with 20 numbers from 0 to 90 (no duplicates). Someone will call out numbers one at a time, and each player will cross out a number on his card as it is called. The first player with all the numbers crossed out is the winner. In the simulation, as the game is in progress, each player's card is displayed on the screen.

The programmer envisions a short driver class whose `main` method has just two statements:

```
BingoGame b = new BingoGame();
b.playBingo();
```

The `BingoGame` class will have several objects: a `Display`, a `Caller`, and a `PlayerGroup`. The `PlayerGroup` will have a list of `Players`, and each `Player` will have a `BingoCard`.

32. The relationship between the `PlayerGroup` and `Player` classes is an example of
    (A) procedural abstraction.
    (B) data encapsulation.
    (C) composition.
    (D) inheritance.
    (E) independent classes.

33. Which is a reasonable data structure for a `BingoCard` object? Recall that there are 20 integers from 0 to 90 on a `BingoCard`, with no duplicates. There should also be mechanisms for crossing off numbers that are called, and for detecting a winning card (i.e., one where all the numbers have been crossed off).

    I
    ```
    int[] bingoCard;   //will contain 20 integers
                       //bingoCard[k] is crossed off by setting it to -1.
    int numCrossedOff;   //player wins when numCrossedOff reaches 20.
    ```

    II
    ```
    boolean[] bingoCard;  //will contain 91 boolean values, of which
                          //20 are true. All the other values are false.
                          //Thus, if bingoCard[k] is true, then k is
                          //on the card, 0 <= k <= 90. A number k is
                          //crossed off by changing the value of
                          //bingoCard[k] to false.
    int numCrossedOff;   //player wins when numCrossedOff reaches 20.
    ```

    III
    ```
    ArrayList<Integer> bingoCard;  //will contain 20 integers.
          //A number is crossed off by removing it from the ArrayList.
          //Player wins when bingoCard.size() == 0.
    ```

    (A) I only
    (B) II only
    (C) III only
    (D) I and II only
    (E) I, II, and III

GO ON TO THE NEXT PAGE.

34. The programmer decides to use an `ArrayList<Integer>` to store the numbers to be called by the `Caller`, but there is an error in the code.

```
public class Caller
{
    private ArrayList<Integer> numbers;

    public Caller()
    {
        numbers = getList();
        shuffleNumbers();
    }

    /** Returns the numbers 0...90 in order. */
    private ArrayList<Integer> getList()
    { /* implementation not shown */ }

    /** Shuffle the numbers. */
    private void shuffleNumbers()
    { /* implementation not shown */ }
}
```

When the programmer tests the constructor of the `Caller` class, she gets a `NullPointerException`. Which could be the cause of this error?

(A) The `Caller` object in the driver class was not created with `new`.

(B) The programmer forgot the `return` statement in `getList` that returns the list of `Integers`.

(C) The declaration of `numbers` is incorrect. It needed to be

```
private ArrayList<Integer> numbers = null;
```

(D) In the `getList` method, an attempt was made to add an `Integer` to an `ArrayList` that had not been created with `new`.

(E) The `shuffleNumbers` algorithm went out of range, causing a `null Integer` to be shuffled into the `ArrayList`.

GO ON TO THE NEXT PAGE.

35. Consider method `findSomething` below.

```
/** Precondition: a.length is equal to b.length. */
public static boolean findSomething(int[] a, int[] b)
{
    for (int aValue: a)
    {
        boolean found = false;
        for (int bValue: b)
        {
            if (bValue == aValue)
                found = true;
        }
        if (!found)
            return false;
    }
    return true;
}
```

Which best describes what method `findSomething` does? Method `findSomething` returns `true` only if

(A) arrays `a` and `b` contain identical elements in the same order.
(B) arrays `a` and `b` contain identical elements in reverse order.
(C) arrays `a` and `b` are permutations of each other.
(D) array `a` contains at least one element that is also in `b`.
(E) every element of array `a` is also in `b`.

GO ON TO THE NEXT PAGE.

36. Consider a program that has a two-dimensional array `mat` of `int` values. The program has several methods that change `mat` by reflecting elements of `mat` across a mirror placed symmetrically on the matrix. Here are five such methods:

`mirrorVerticalLeftToRight`	transforms	2 4 6 1 3 5 8 9 0	to	2 4 2 1 3 1 8 9 8
`mirrorVerticalRightToLeft`	transforms	2 4 6 1 3 5 8 9 0	to	6 4 6 5 3 5 0 9 0
`mirrorHorizontalTopToBottom`	transforms	2 4 6 1 3 5 8 9 0	to	2 4 6 1 3 5 2 4 6
`mirrorHorizontalBottomToTop`	transforms	2 4 6 1 3 5 8 9 0	to	8 9 0 1 3 5 8 9 0
`mirrorDiagonalRightToLeft`	transforms	2 4 6 1 3 5 8 9 0	to	2 4 6 4 3 5 6 5 0

Consider the following method that transforms the matrix in one of the ways shown above.

```
public static void someMethod(int[][] mat)
{
    int height = mat.length;
    int numCols = mat[0].length;
    for (int col = 0; col < numCols; col++)
        for (int row = 0; row < height/2; row++)
            mat[height - row - 1][col] = mat[row][col];
}
```

Which method described above corresponds to `someMethod`?

(A) `mirrorVerticalLeftToRight`
(B) `mirrorVerticalRightToLeft`
(C) `mirrorHorizontalTopToBottom`
(D) `mirrorHorizontalBottomToTop`
(E) `mirrorDiagonalRightToLeft`

GO ON TO THE NEXT PAGE.

Refer to the following for Questions 37 and 38.

A word creation game uses a set of small letter tiles, all of which are initially in a tile bag. A partial implementation of a `TileBag` class is shown below.

```
public class TileBag
{
    //tiles contains all the tiles in the bag
    private List<Tile> tiles;
    //size is the number of not-yet-used tiles
    private int size;

    //Constructors and other methods are not shown.
}
```

Consider the following method in the `TileBag` class that allows a player to get a new tile from the `TileBag`.

```
public Tile getNewTile()
{
    if (size == 0)  //no tiles left
        return null;
    int index = (int) (Math.random() * size);
    size--;
    Tile temp = tiles.get(index);
    /* code to swap tile at position size with tile at position index */
    return temp;
}
```

37. Which `/* code to swap tile at position size with tile at position index */` performs the swap correctly?

    (A)
    ```
    tiles.set(size, temp);
    tiles.set(index, tiles.get(size));
    ```

    (B)
    ```
    tiles.set(index, tiles.get(size));
    tiles.set(size, temp);
    ```

    (C)
    ```
    tiles.swap(index, size);
    ```

    (D)
    ```
    tiles.get(size, temp);
    tiles.get(index, tiles.set(size));
    ```

    (E)
    ```
    tiles.get(index, tiles.set(size));
    tiles.get(size, temp);
    ```

38. Which is true about the `getNewTile` algorithm?

    (A) The algorithm allows the program to keep track of both used and unused tiles.

    (B) The `tiles` list becomes one element shorter when `getNewTile` is executed.

    (C) The algorithm selects a random `Tile` from all tiles in the list.

    (D) The `tiles` list has used tiles in the beginning and unused tiles at the end.

    (E) The `tiles` list contains only tiles that have not been used.

GO ON TO THE NEXT PAGE.

39. Consider the following two classes.

```
public class Bird
{
    public void act()
    {
        System.out.print("fly ");
        makeNoise();
    }

    public void makeNoise()
    {
        System.out.print("chirp ");
    }
}

public class Dove extends Bird
{
    public void act()
    {
        super.act();
        System.out.print("waddle ");
    }

    public void makeNoise()
    {
        super.makeNoise();
        System.out.print("coo ");
    }
}
```

Suppose the following declaration appears in a class other than `Bird` or `Dove`.

```
Bird pigeon = new Dove();
```

What is printed as a result of the call `pigeon.act()`?

(A) `fly`
(B) `fly chirp`
(C) `fly chirp waddle`
(D) `fly chirp waddle coo`
(E) `fly chirp coo waddle`

GO ON TO THE NEXT PAGE.

40. Consider a method `partialProd` that returns an integer array `prod` such that for all `k`, `prod[k]` is equal to `arr[0] * arr[1] * ⋯ arr[k]`. For example, if `arr` contains the values `{2,5,3,4,10}`, the array `prod` will contain the values `{2,10,30,120,1200}`.

```
public static int[] partialProd(int[] arr)
{
    int[] prod = new int[arr.length];
    for (int j = 0; j < arr.length; j++)
        prod[j] = 1;
    /* missing code */
    return prod;
}
```

Consider the following two implementations of `/* missing code */`.

Implementation 1

```
for (int j = 1; j < arr.length; j++)
{
    prod[j] = prod[j - 1] * arr[j];
}
```

Implementation 2

```
for (int j = 0; j < arr.length; j++)
    for (int k = 0; k <= j; k++)
    {
        prod[j] = prod[j] * arr[k];
    }
```

Which of the following statements is true?

(A) Both implementations work as intended but Implementation 1 is faster than Implementation 2.

(B) Both implementations work as intended but Implementation 2 is faster than Implementation 1.

(C) Both implementations work as intended and are equally fast.

(D) Implementation 1 doesn't work as intended because the elements of `prod` are incorrectly assigned.

(E) Implementation 2 doesn't work as intended because the elements of `prod` are incorrectly assigned.

END OF SECTION I

COMPUTER SCIENCE A
SECTION II

Time—1 hour and 30 minutes
Number of questions—4
Percent of total grade—50

DIRECTIONS: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Write your answers in pencil only in the booklet provided.

NOTES:

- Assume that the classes in the Quick Reference have been imported where needed.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

GO ON TO THE NEXT PAGE.

1. In this question you will write two methods of an `Experiment` class that handles chemical solutions.

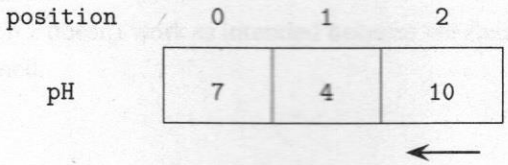
   A chemical solution is said to be acidic if it has a pH integer value from 1 to 6 inclusive. The lower the pH, the more acidic the solution.

   An experiment has three chemical solutions, numbered 0, 1, and 2, arranged in a line, and a mechanical arm that moves back and forth along the line, stopping at any of the solutions.

   A chemical solution is specified by a `Solution` class shown as follows.

```
public class Solution
{
    private int PH;
    private int positionLabel;

    /** Returns the PH of the solution, an
     * integer ranging from 1 (very acidic) to 14.
     */
    int getPH()
    { return PH; }

    /** Returns the position label of the solution, an
     * integer ranging from 0 to 2, inclusive.
     */
    int getPos()
    { return positionLabel; }

    //Constructors and other methods not shown.
}
```

   The experiment keeps track of the solutions and the mechanical arm. The figure below represents the solutions and mechanical arm in an experiment. The arm, indicated by the arrow, is currently at the solution whose position is at 2. The integers in the boxes represent the pH values of the solutions.



   In this experiment, the most acidic solution is at position 1, since its pH value is the lowest. The mechanical arm, which is specified by a position and a direction, is at position 2 and facing left.

GO ON TO THE NEXT PAGE.

The `MechanicalArm` class is shown below.

```
public class MechanicalArm
{
    private int currentPos;
    private boolean facesRight;

    /** Returns the current position of the mechanical arm.
     */
    public int getCurrentPos()
    { return currentPos; }

    /** Returns true if the mechanical arm is facing
     *  right (toward higher position numbers);
     *  false if it is facing left.
     */
    public boolean isFacingRight()
    { /* implementation not shown */ }

    /** Changes direction of the mechanical arm.
     */
    public void changeDirection()
    { /* implementation not shown */ }

    /** Moves the mechanical arm forward numLocs
     *  positions in its current direction.
     *  Precondition: numLocs >= 0.
     *  Postcondition: 0 <= currentPos <=2.
     */
    public void moveForward (int numLocs)
    { /* implementation not shown */ }

    //Constructors and other methods not shown.
}
```

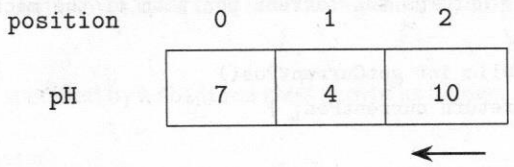
An experiment is represented by the `Experiment` class shown below.

```
public class Experiment
{
    private MechanicalArm arm;
    private Solution s0, s1, s2;

    /** Resets the experiment, such that the mechanical arm has
     *  a current position of 0 and is facing right.
     */
    public void reset()
    { /* to be implemented in part(a) */ }

    /** Returns the position of the most acidic solution,
     *  or -1 if there are no acidic solutions, as
     *  described in part(b).
     */
    public int mostAcidic()
    { /* to be implemented in part(b) */ }
}
```

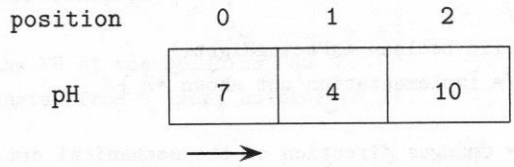
GO ON TO THE NEXT PAGE.

(a) Write the `Experiment` method `reset` that places the mechanical arm facing right, at index `0`.

For example, suppose the experiment contains the solutions with pH values shown. The arrow represents the mechanical arm.



A call to `reset` will result in



Class information for this question

```
public class Solution

private int PH
private int positionLabel
int getPH()
int getPos()

public class MechanicalArm

private int currentPos
private boolean facesRight
int getCurrentIndex()
boolean isFacingRight()
void changeDirection()
void moveForward(int numLocs)

public class Experiment

private MechanicalArm arm
private Solution s0, s1, s2
public void reset()
public int mostAcidic()
```

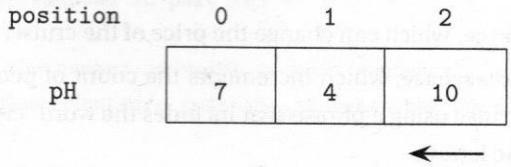
GO ON TO THE NEXT PAGE.

Complete method `reset` below.

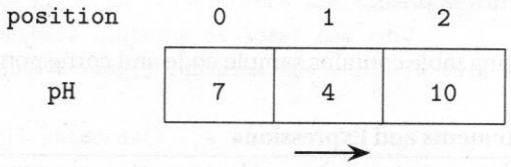
```
/** Resets the experiment, such that the mechanical arm has
 *  a current position of 0 and is facing right.
 */
public void reset()
```

(b) Write the `Experiment` method `mostAcidic` that returns the `position` of the most acidic solution and places the mechanical arm facing right at the location of the most acidic solution. A solution is acidic if its pH is less than 7. The lower the pH, the more acidic the solution. If there are no acidic solutions in the experiment, the `mostAcidic` method should return `-1`, and place the mechanical arm at position `0`, facing right.

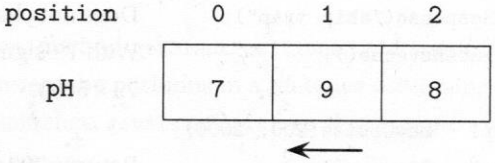
For example, suppose the experiment has this state:



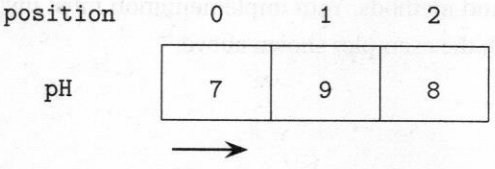
A call to `mostAcidic` should return the value `1` and result in the following state for the experiment:



If the experiment has this state,



a call to `mostAcidic` should return the value `-1` and result in the following state for the experiment:



GO ON TO THE NEXT PAGE.

Complete method `mostAcidic` below.

```
/** Returns the position of the most acidic solution,
 *  or -1 if there are no acidic solutions, as
 *  described in part(b).
 */
public int mostAcidic()
```

2. This question involves keeping track of details for a world cruise, using a `Cruise` class. A `Cruise` object is created with two parameters, the number of people currently signed up for the cruise, and the current price.

   The `Cruise` class provides a constructor and the following methods.

   - `setPrice`, which can change the price of the cruise.
   - `checkResponse`, which increments the count of people if someone has requested the cruise using a phrase that includes the word "cruise". You may assume all lowercase letters.
   - `calculateRevenue`, which returns the number of signups so far multiplied by the current price. Note that if more than 300 people have signed up for the cruise, everyone will receive a $500 discount off the current price. If between 200 and 300 (including 200) people have signed up, everyone will receive a $350 discount off the current price.

   The following table contains sample code and corresponding results.

Statements and Expressions	Comments
`Cruise cr = newCruise(78, 4000);`	There are 78 signups so far, and the cruise price is $4,000.
`cr.setPrice(5000);`	Changes the price to $5,000.
`cr.checkResponse("world cruise");`	Increments signup number to 79.
`cr.checkResponse("ship trip");`	Does not change number. It is still 78.
`cr.calculateRevenue();`	With 79 signups and price $5,000, returns 79 × 5000.
`Cruise cr1 = newCruise(200, 2000);`	
`cr1.calculateRevenue();`	Returns 200 × 1650
`Cruise cr2 = newCruise(397, 6000);`	
`cr2.calculateRevenue();`	Returns 397 × 5500

Write the complete `Cruise` class, including the constructor and any required instance variables and methods. Your implementation must meet all specifications and be consistent with the examples shown above.

GO ON TO THE NEXT PAGE.

3. A text-editing program uses a `Sentence` class that manipulates a single sentence. A sentence contains letters, blanks, and punctuation. The first character in a sentence is a letter, and the last character is a punctuation mark. Any two words in the sentence are separated by a single blank. A partial implementation of the `Sentence` class is as follows.

```
public class Sentence
{
    /** The sentence to manipulate */
    private String sentence;

    /**  Returns an ArrayList of integer positions containing a
     *   blank in this sentence. If there are no blanks in the
     *   sentence, returns an empty list.
     */
    public ArrayList<Integer> getBlankPositions()
    { /* to be implemented in part (a) */ }

    /**  Returns the number of words in this sentence
     *   Precondition: Sentence contains at least one word.
     */
    public int countWords()
    { /* implementation not shown */ }

    /** Returns the array of words in this sentence.
     *  Precondition:
     *   - Any two words in the sentence are separated by one blank.
     *   - The sentence contains at least one word.
     *  Postcondition: String[] contains the words in this sentence.
     */
    public String[] getWords()
    { /* to be implemented in part (b) */ }

    //Constructor and other methods are not shown.
}
```

(a) Write the `Sentence` method `getBlankPositions`, which returns an `ArrayList` of integers that represent the positions in a sentence containing blanks. If there are no blanks in the sentence, `getBlankPositions` should return an empty list.

Some results of calling `getBlankPositions` are shown below.

Sentence	Result of call to `getBlankPositions`
`I love you!`	`[1, 6]`
`The cat sat on the mat.`	`[3, 7, 11, 14, 18]`
`Why?`	`[ ]`

GO ON TO THE NEXT PAGE.

Complete method `getBlankPositions` below.

```
/**  Returns an ArrayList of integer positions containing a
 *   blank in this sentence. If there are no blanks in the
 *   sentence, returns an empty list.
 */
public ArrayList<Integer> getBlankPositions()
```

(b) Write the `Sentence` method `getWords`, which returns an array of words in the sentence. A word is defined as a string of letters and punctuation, and does not contain any blanks. You may assume that a sentence contains at least one word.

Some examples of calling `getWords` are shown below.

Sentence	Result returned by `getWords`
`The bird flew away.`	`{The, bird, flew, away.}`
`Wow!`	`{Wow!}`
`Hi!  How are you?`	`{Hi!, How, are, you?}`

In writing method `getWords`, you *must* use methods `getBlankPositions` and `countWords`.

Complete method `getWords` below.

```
/** Returns the array of words in this sentence.
 *  Precondition:
 *   - Any two words in the sentence are separated by one blank.
 *   - The sentence contains at least one word.
 *  Postcondition: String[] contains the words in this sentence.
 */
public String[] getWords()
```

4. This question manipulates a two-dimensional array. In parts (a) and (b) you will write two methods of a `Matrix` class.

In doing so, you will use the `reverseArray` method shown in the class below.

```
public class ArrayUtil
{
    /** Reverses elements of array arr.
     *  Precondition:  arr.length > 0.
     *  Postcondition: The elements of arr have been reversed.
     */
    public static void reverseArray(int[] arr)
    { /* implementation not shown */ }

    //Other methods are not shown.
}
```

GO ON TO THE NEXT PAGE.

Consider the following incomplete `Matrix` class, which represents a two-dimensional matrix of integers. Assume that the matrix contains at least one integer.

```
public class Matrix
{
    private int[][] mat;

    /** Constructs a matrix of integers. */
    public Matrix (int[][] m)
    {  mat = m; }

    /** Reverses the elements in each row of mat.
     *  Postcondition: The elements in each row have been reversed.
     */
    public void reverseAllRows()
    { /* to be implemented in part (a) */ }

    /** Reverses the elements of mat, as described in part (b).
     */
    public void reverseMatrix()
    { /* to be implemented in part (b) */ }

    //Other instance variables, constructors and methods are not shown.
}
```

(a) Write the `Matrix` method `reverseAllRows`. This method reverses the elements of each row. For example, if `mat1` refers to a `Matrix` object, then the call `mat1.reverseAllRows()` will change the matrix as shown below.

Before call

	0	1	2	3
0	1	2	3	4
1	5	6	7	8
2	9	10	11	12

After call

	0	1	2	3
0	4	3	2	1
1	8	7	6	5
2	12	11	10	9

In writing `reverseAllRows`, you *must* call the `reverseArray` method in the `ArrayUtil` class.

Complete method `reverseAllRows` below.

```
/** Reverses the elements in each row of mat.
 *  Postcondition: The elements in each row have been reversed.
 */
public void reverseAllRows()
```

GO ON TO THE NEXT PAGE.

(b) Write the `Matrix` method `reverseMatrix`. This method reverses the elements of a matrix such that the final elements of the matrix, when read in row-major order, are the same as the original elements when read from the bottom corner, right to left, going upward. Again let `mat1` be a reference to a `Matrix` object. The call `mat1.reverseMatrix()` will change the matrix as shown below.

Before call

	0	1
0	1	2
1	3	4
2	5	6

After call

	0	1
0	6	5
1	4	3
2	2	1

In writing `reverseMatrix`, you *must* call the `reverseAllRows` method in part (a). Assume that `reverseAllRows` works correctly regardless of what you wrote in part (a).

Complete method `reverseMatrix` below.

```
/** Reverses the elements of mat, as described in part (b).
 */
public void reverseMatrix()
```

STOP

END OF EXAM