

# Practice Test 2

## COMPUTER SCIENCE A

### SECTION I

**Time—1 hour and 30 minutes**

**40 Questions**

**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.

**PRACTICE TEST 2**

- What output is produced by the following line of code?

```
System.out.println("\\\"This is\\n very strange\\\"");
```

- (A) \This is\n very strange\
- (B) "This is very strange"
- (C) This is  
very strange
- (D) \"This is  
very strange\"
- (E) "This is  
very strange"

**PRACTICE TEST 2**

2. A certain class, `SomeClass`, contains a method with the following header.

```
public int getValue(int n)
```

Suppose that methods with the following headers are now added to `SomeClass`.

- I. public int getValue()
- II. public double getValue(int n)
- III. public int getValue(double n)

Which of the above headers will cause an error?

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

3. Consider the following statement.

```
int num = /* expression */;
```

Which of the following replacements for `/* expression */` creates in `num` a random integer from 2 to 50, including 2 and 50?

- (A) `(int)(Math.random() * 50) - 2`
- (B) `(int)(Math.random() * 49) - 2`
- (C) `(int)(Math.random() * 49) + 2`
- (D) `(int)(Math.random() * 50) + 2`
- (E) `(int)(Math.random() * 48) + 2`

4. Consider the following code segment.

```
int num = 0, score = 10;
if (num != 0 && score / num > SOME_CONSTANT)
    statement1;
else
    statement2;
```

What is the result of executing this statement?

- (A) An `ArithmeticException` will be thrown.
- (B) A syntax error will occur.
- (C) `statement1`, but not `statement2`, will be executed.
- (D) `statement2`, but not `statement1`, will be executed.
- (E) Neither `statement1` nor `statement2` will be executed; control will pass to the first statement following the `if` statement.

5. The following shuffle algorithm is used to shuffle an array of int values, nums.

```
public void shuffle ()
{
    for (int k = nums.length - 1; k > 0; k--)
    {
        int randPos = (int) (Math.random() * (k + 1));
        int temp = nums[k];
        nums[k] = nums[randPos];
        nums[randPos] = temp;
    }
}
```

Suppose the initial state of nums is 8, 7, 6, 5, 4, and when the method is executed the values generated for randPos are 3, 2, 0, 0, in that order. What element will be contained in nums[2] after execution?

- (A) 8  
 (B) 7  
 (C) 6  
 (D) 5  
 (E) 4
6. Consider the following instance variables and method assignValues in the same class.

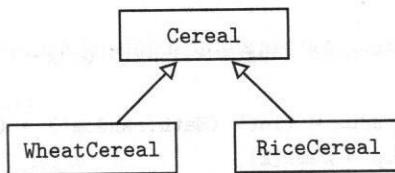
```
private int numRows;
private int numCols;
private int[][] mat;

/** arr has numCols elements */
private void assignValues(int[] arr, int value)
{
    for (int k = 0; k < arr.length; k++)
        arr[k] = value;
}
```

Which of the following code segments will correctly assign mat to have the value 100 in each slot? You may assume that the instance variables have all been correctly initialized.

- I. for (int row = 0; row < numRows; row++)
 assignValues(mat[row], 100);
  - II. for (int col = 0; col < numCols; col++)
 assignValues(mat[col], 100);
  - III. for (int[] row: mat)
 for (int num: row)
 num = 100;
- (A) I only  
 (B) II only  
 (C) III only  
 (D) I and II only  
 (E) I and III only

7. Consider the following inheritance hierarchy.



Which of the following declarations will not cause an error? You may assume that each of the classes above has a no-argument constructor.

- I. WheatCereal w = new Cereal();
  - II. Cereal c1 = new Cereal();
  - III. Cereal c2 = new RiceCereal();
- (A) I only  
(B) II only  
(C) III only  
(D) II and III only  
(E) I, II, and III

Questions 8 and 9 refer to the following class definitions.

```
public class Class1
{
    public void method1()
    { /* implementation not shown */ }
}

public class Class2 extends Class1
{
    public void method2()
    { /* implementation not shown */ }

    //Private instance variables and other methods are not shown.
}

public class Class3 extends Class2
{
    public void method3(Class3 other)
    { /* implementation not shown */ }

    //Private instance variables and other methods are not shown.
}
```

8. Assuming that Class1, Class2, and Class3 have no-argument constructors, which is (are) valid in a client class?

- I. Class1 c1 = new Class2();
  - II. Class2 c2 = new Class3();
  - III. Class1 c3 = new Class3();
- (A) I only  
 (B) II only  
 (C) III only  
 (D) I and II only  
 (E) I, II, and III

9. Consider the following declarations in a client class.

```
Class3 ob3 = new Class3();
Class2 ob2 = new Class2();
```

Which method calls would be legal?

- I. ob3.method1();
  - II. ob2.method3(ob3);
  - III. ob3.method3(ob2);
- (A) I only  
 (B) II only  
 (C) III only  
 (D) II and III only  
 (E) I, II, and III

10. Refer to the following program segment.

```
for (int n = 50; n > 0; n = n / 2)
    System.out.println(n);
```

How many lines of output will this segment produce?

- (A) 50
- (B) 49
- (C) 7
- (D) 6
- (E) 5

11. Let `list` be an `ArrayList<String>` containing only these elements.

```
"John", "Mary", "Harry", "Luis"
```

Which of the following statements will cause an error to occur?

- I. `list.set(2, "6");`
  - II. `list.add(4, "Pat");`
  - III. `String s = list.get(4);`
- (A) I only
  - (B) II only
  - (C) III only
  - (D) II and III only
  - (E) I, II, and III

12. Consider the following static method.

```
public static int compute(int n)
{
    for (int i = 1; i < 4; i++)
        n *= n;
    return n;
}
```

Which of the following could replace the body of `compute`, so that the new version returns the identical result as the original for all `n`?

- (A) `return 4 * n;`
- (B) `return 8 * n;`
- (C) `return 64 * n;`
- (D) `return (int) Math.pow(n, 4);`
- (E) `return (int) Math.pow(n, 8);`

13. Consider the following instance variable and method.

```
private int[] nums;
/** Post-condition: If nums contains odd values, then
 * return the index of the first odd value. If all values
 * are even, then return -1. If nums is empty, then return -1.
 */
public int getValue()
{
    for (int k = 0; k < nums.length; k++)
        if (nums[k] % 2 != 0)
            return k;
    return -1;
}
```

Suppose the following statement is executed.

```
int j = getValue();
```

If the value returned in *j* is a positive integer, which of the following best describes the contents of *nums*?

- (A) The only odd int in *nums* is at position *j*.
- (B) All values in positions 0 through *j*-1 are odd.
- (C) All values in positions 0 through *j*-1 are even.
- (D) All values in positions *nums.length*-1 down to *j*+1 are odd.
- (E) All values in positions *nums.length*-1 down to *j*+1 are even.

14. Consider the following method.

```
public int mystery (int n)
{
    if (n == 0)
        return 0;
    else if (n % 2 == 1)
        return n;
    else
        return n + mystery(n - 1);
}
```

What will be returned by a call to *mystery(6)*?

- (A) 6
- (B) 11
- (C) 12
- (D) 27
- (E) 30

15. Consider the following code segment.

```
int num1 = value1, num2 = value2, num3 = value3;
while (num1 > num2 || num1 > num3)
{
    /* body of loop */
}
```

You may assume that `value1`, `value2`, and `value3` are `int` values. Which of the following is sufficient to guarantee that `/* body of loop */` will never be executed?

- (A) There is no statement in `/* body of loop */` that leads to termination.
- (B) `num1 < num2`
- (C) `num1 < num3`
- (D) `num1 > num2 && num1 > num3`
- (E) `num1 < num2 && num1 < num3`

16. Consider the following two classes.

```
public class Performer
{
    public void act()
    {
        System.out.print(" bow");
        perform();
    }

    public void perform()
    {
        System.out.print(" act");
    }
}

public class Singer extends Performer
{
    public void act()
    {
        System.out.print(" rise");
        super.act();
        System.out.print(" encore");
    }

    public void perform()
    {
        System.out.print(" aria");
    }
}
```

Suppose the following declaration appears in a class other than Performer or Singer.

```
Performer p = new Singer();
```

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

- (A) `rise bow aria encore`
- (B) `rise bow act encore`
- (C) `rise bow act`
- (D) `bow act aria`
- (E) `bow aria encore`

Use the program description below for Questions 17–19.

A car dealer needs a program that will maintain an inventory of cars on his lot. There are four types of cars: sedans, station wagons, electric cars, and SUVs. The model, year, color, and price need to be recorded for each car, plus any additional features for the different types of cars. The program must allow the dealer to

- Add a new car to the lot.
- Remove a car from the lot.
- Correct any data that have been entered.
- Display information for any car.

17. The programmer decides to have these classes: Car, Inventory, Sedan, SUV, ElectricCar, and StationWagon. Which statement is true about the relationships between these classes and their attributes?

- I. There are no inheritance relationships between these classes.
- II. The Inventory class *has-a* list of Car objects.
- III. The Sedan, SUV, ElectricCar, and StationWagon classes are independent of each other.
  - (A) I only
  - (B) II only
  - (C) III only
  - (D) I and II only
  - (E) II and III only

18. Suppose that the programmer decides to have a Car class and an Inventory class. The Inventory class will maintain a list of all the cars on the lot. Here are some of the methods in the program.

```
addCar          //adds a car to the lot
removeCar       //removes a car from the lot
displayCar      //displays all the features of a given car
setColor        //sets the color of a car to a given color
                  // (may be used to correct data)
getPrice        //returns the price of a car
displayAllCars  //displays features for every car on the lot
```

In each of the following, a class and a method are given. Which is the least suitable choice of class to be responsible for the given method?

- (A) Car, setColor
- (B) Car, removeCar
- (C) Car, getPrice
- (D) Car, displayCar
- (E) Inventory, displayAllCars

19. Suppose Car is a superclass and Sedan, StationWagon, ElectricCar, and SUV are subclasses of Car. Which of the following is the most likely method of the Car class to be overridden by at least one of the subclasses (Sedan, StationWagon, ElectricCar, or SUV)?

- (A) `setColor(newColor)` //sets color of Car to newColor
- (B) `getModel()` //returns model of Car
- (C) `displayCar()` //displays all features of Car
- (D) `setPrice(newPrice)` //sets price of Car to newPrice
- (E) `getYear()` //returns year of Car

20. Consider the following segment of code.

```
String word = "conflagration";
int x = word.indexOf("flag");
String s = word.substring(0, x);
```

What will be the result of executing the above segment?

- (A) A syntax error will occur.
- (B) String s will be the empty string.
- (C) String s will contain "flag".
- (D) String s will contain "conf".
- (E) String s will contain "con".

21. A two-dimensional matrix `mat` with at least one row is initialized and will be traversed using a row-major (row-by-row, left-to-right) traversal. Which represents the last element accessed?

- (A) `mat[mat.length][mat[0].length]`
- (B) `mat[mat[0].length][mat.length]`
- (C) `mat[mat.length - 1][mat[0].length - 1]`
- (D) `mat[mat[0].length - 1][mat.length - 1]`
- (E) `mat[mat.length - 1][mat.length - 1]`

22. A class of 30 students rated their computer science teacher on a scale of 1 to 10 (1 means awful and 10 means outstanding). The `responses` array is a 30-element integer array of the student responses. An 11-element array `freq` will count the number of occurrences of each response. For example, `freq[6]` will count the number of students who responded 6. The quantity `freq[0]` will not be used.

Here is a program that counts the students' responses and outputs the results.

```
public class StudentEvaluations
{
    public static void main(String args[])
    {
        int[] responses = {6,6,7,8,10,1,5,4,6,7,
                           5,4,3,4,4,9,8,6,7,10,
                           6,7,8,8,9,6,7,8,9,2};
        int[] freq = new int[11];
        for (int i = 0; i < responses.length; i++)
            freq[responses[i]]++;
        //output results
        System.out.print("rating" + " " + "frequency\n");
        for (int rating = 1; rating < freq.length; rating++)
            System.out.print(rating + " " +
                            freq[rating] + "\n");
    }
}
```

Suppose the last entry in the initializer list for the `responses` array was incorrectly typed as 12 instead of 2. What would be the result of running the program?

- (A) A rating of 12 would be listed with a frequency of 1 in the output table.
- (B) A rating of 1 would be listed with a frequency of 12 in the output table.
- (C) An `ArrayIndexOutOfBoundsException` would be thrown.
- (D) A `StringIndexOutOfBoundsException` would be thrown.
- (E) A `NullPointerException` would be thrown.

Questions 23–25 are based on the three classes below.

```
public class Employee
{
    private String name;
    private int employeeNum;
    private double salary, taxWithheld;

    public Employee(String aName, int empNum, double aSalary,
                    double aTax)
    { /* implementation not shown */ }

    /** @return pre-tax salary */
    public double getSalary()
    { return salary; }

    public String getName()
    { return name; }

    public int getEmployeeNum()
    { return employeeNum; }

    public double getTax()
    { return taxWithheld; }

    public double computePay()
    { return salary - taxWithheld; }
}

public class PartTimeEmployee extends Employee
{
    private double payFraction;

    public PartTimeEmployee(String aName, int empNum, double aSalary,
                           double aTax, double aPayFraction)
    { /* implementation not shown */ }

    public double getPayFraction()
    { return payFraction; }

    public double computePay()
    { return getSalary() * payFraction - getTax(); }
}

public class Consultant extends Employee
{
    private static final double BONUS = 5000;

    public Consultant(String aName, int empNum, double aSalary,
                      double aTax)
    { /* implementation not shown */ }

    public double computePay()
    { /* implementation code */ }
}
```

23. The `computePay` method in the `Consultant` class redefines the `computePay` method of the `Employee` class to add a bonus to the salary after subtracting the tax withheld. Which represents correct /\* *implementation code* \*/ of `computePay` for `Consultant`?

- I. `return super.computePay() + BONUS;`
  - II. `super.computePay();  
return getSalary() + BONUS;`
  - III. `return getSalary() - getTax() + BONUS;`
- (A) I only  
 (B) II only  
 (C) III only  
 (D) I and III only  
 (E) I and II only

24. Consider these valid declarations in a client program.

```
Employee e = new Employee("Noreen Rizvi", 304, 65000, 10000);
Employee p = new PartTimeEmployee("Rafael Frongillo", 287, 40000,
    7000, 0.8);
Employee c = new Consultant("Dan Lepage", 694, 55000, 8500);
```

Which of the following method calls will cause an error?

- (A) `double x = e.computePay();`  
 (B) `double y = p.computePay();`  
 (C) `String n = c.getName();`  
 (D) `int num = p.getEmployeeNum();`  
 (E) `double g = p.getPayFraction();`

25. Consider the `writePayInfo` method.

```
/** Writes Employee name and pay on one line. */
public static void writePayInfo(Employee e)
{ System.out.println(e.getName() + " " + e.computePay()); }
```

The following piece of code invokes this method.

```
Employee[] empList = new Employee[3];
empList[0] = new Employee("Lila Fontes", 1, 10000, 850);
empList[1] = new Consultant("Momo Liu", 2, 50000, 8000);
empList[2] = new PartTimeEmployee("Moses Wilks", 3, 25000, 3750,
    0.6);
for (Employee e : empList)
    writePayInfo(e);
```

What will happen when this code is executed?

- (A) A `NullPointerException` will be thrown.
- (B) An `ArrayIndexOutOfBoundsException` will be thrown.
- (C) A compile-time error will occur, with the message that the `getName` method is not in the `Consultant` class.
- (D) A compile-time error will occur, with the message that an instance of an `Employee` object cannot be created.
- (E) A list of employees' names and corresponding pay will be written to the screen.

26. Consider an array `arr` that is initialized with `int` values. The following code segment stores in `count` the number of positive values in `arr`.

```
int count = 0, index = 0;
while (index < arr.length)
{
    if (arr[index] > 0)
        count++;
    index++;
}
```

Which of the following is equivalent to the above segment?

- I. 

```
int count = 0;
for (int num : arr)
{
    if (arr[num] > 0)
        count++;
}
```
- II. 

```
int count = 0;
for (int num : arr)
{
    if (num > 0)
        count++;
}
```
- III. 

```
int count = 0;
for (int i = 0; i < arr.length; i++)
{
    if (arr[i] > 0)
        count++;
}
```

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

27. A square matrix is declared as

```
int[][] mat = new int[SIZE][SIZE];
```

where SIZE is an appropriate integer constant. Consider the following method.

```
public static void mystery(int[][] mat, int value, int top, int left,
    int bottom, int right)
{
    for (int i = left; i <= right; i++)
    {
        mat[top][i] = value;
        mat[bottom][i] = value;
    }
    for (int i = top + 1; i <= bottom - 1; i++)
    {
        mat[i][left] = value;
        mat[i][right] = value;
    }
}
```

Assuming that there are no out-of-range errors, which best describes what method `mystery` does?

- (A) Places value in corners of the rectangle with corners `(top, left)` and `(bottom, right)`
- (B) Places value in the diagonals of the square with corners `(top, left)` and `(bottom, right)`
- (C) Places value in each element of the rectangle with corners `(top, left)` and `(bottom, right)`
- (D) Places value in each element of the border of the rectangle with corners `(top, left)` and `(bottom, right)`
- (E) Places value in the topmost and bottommost rows of the rectangle with corners `(top, left)` and `(bottom, right)`

28. Consider the following declaration.

```
ArrayList<Integer> list = new ArrayList<Integer>();
```

Which of the following code segments will place the integers 1 to 10, in any order, in the empty list?

- I. 

```
for (int i = 0; i < 10; i++)
    list.add(i + 1);
```
- II. 

```
for (int i = 0; i < 10; i++)
    list.add(i, i + 1);
```
- III. 

```
for (int i = 9; i >= 0; i--)
    list.add(i, i + 1);
```

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

29. Assume that a Book class has a `compareTo` method in which, if `b1` and `b2` are Book objects, `b1.compareTo(b2)` is a negative integer if `b1` is less than `b2`, a positive integer if `b1` is greater than `b2`, and 0 if `b1` equals `b2`. The following method is intended to return the index of the “smallest” book—namely, the book that would appear first in a sorted list of Book objects.

```
/** Precondition:
 *  - books is initialized with Book objects.
 *  - books.length > 0.
 */
public static int findMin(Book[] books)
{
    int minPos = 0;
    for (int index = 1; index < books.length; index++)
    {
        if ( /* condition */ )
        {
            minPos = index;
        }
    }
    return minPos;
}
```

Which of the following should be used to replace `/* condition */` so that `findMin` works as intended?

- (A) `books[index] < books[minPos]`
- (B) `books[index] > books[minPos]`
- (C) `books[index].compareTo(books[minPos]) > 0`
- (D) `books[index].compareTo(books[minPos]) >= 0`
- (E) `books[index].compareTo(books[minPos]) < 0`

30. Refer to the static method `removeNegs` shown below.

```
/** Precondition: list is an ArrayList<Integer>.  
 * Postcondition: All negative values have been removed from list.  
 */  
public static void removeNegs(ArrayList<Integer> list)  
{  
    int index = 0;  
    while (index < list.size())  
    {  
        if (list.get(index).intValue() < 0)  
        {  
            list.remove(index);  
        }  
        index++;  
    }  
}
```

For which of the following lists will the method not work as intended?

- (A) 6 -1 -2 5
- (B) -1 2 -3 4
- (C) 2 4 6 8
- (D) -3
- (E) 1 2 3 -8

31. A sorted list of 120 integers is to be searched to determine whether the value 100 is in the list. Assuming that the most efficient searching algorithm is used, what is the maximum number of elements that must be examined?
- (A) 7  
(B) 8  
(C) 20  
(D) 100  
(E) 120
32. Consider a sorted array `arr` of  $n$  elements, where  $n$  is large and  $n$  is even. Under which conditions will a sequential search of `arr` be faster than a binary search?
- I. The target is not in the list.  
II. The target is in the first position of the list.  
III. The target is in `arr[1 + n/2]`.
- (A) I only  
(B) II only  
(C) III only  
(D) I and III only  
(E) II and III only

33. Refer to the following data field and method.

```
private int[] arr;

/** Precondition: arr.length > 0 and index < arr.length. */
public void remove(int index)
{
    int[] b = new int[arr.length - 1];
    int count = 0;
    for (int i = 0; i < arr.length; i++)
    {
        if (i != index)
        {
            b[count] = arr[i];
            count++;
        }
    }
    /* assertion */
    arr = b;
}
```

Which of the following assertions is true when the **/\* assertion \*/** line is reached during execution of **remove**?

- (A)  $b[k] == arr[k]$  for  $0 \leq k < arr.length$ .
- (B)  $b[k] == arr[k + 1]$  for  $0 \leq k < arr.length$ .
- (C)  $b[k] == arr[k]$  for  $0 \leq k \leq index$ , and  
 $b[k] == arr[k + 1]$  for  $index < k < arr.length - 1$ .
- (D)  $b[k] == arr[k]$  for  $0 \leq k < index$ , and  
 $b[k] == arr[k + 1]$  for  $index \leq k < arr.length - 1$ .
- (E)  $b[k] == arr[k]$  for  $0 \leq k < index$ , and  
 $b[k] == arr[k + 1]$  for  $index \leq k < arr.length$ .

34. Consider the following code segment.

```
for (int n = 25; n >= 0; n /= 2)
    System.out.println(n);
```

When the segment is executed, how many passes through the **for** loop will there be?

- (A) Fewer than 5
- (B) Between 5 and 12, inclusive
- (C) Between 13 and 25, inclusive
- (D) Between 26 and 100, inclusive
- (E) More than 100

Questions 35–37 refer to the TennisPlayer, GoodPlayer, and WeakPlayer classes below. These classes are to be used in a program to simulate a game of tennis.

```

public class TennisPlayer
{
    private String name;

    public TennisPlayer(String aName)
    { name = aName; }

    public boolean serve()
    { /* implementation not shown */ }
}

public class GoodPlayer extends TennisPlayer
{
    public GoodPlayer(String aName)
    { /* implementation not shown */ }

    public boolean serve()
    { /* implementation not shown */ }
}

public class WeakPlayer extends TennisPlayer
{
    public WeakPlayer(String aName)
    { /* implementation not shown */ }

    /** Returns true if serve is in (45% probability),
     *  false if serve is out (55% probability).
     */
    public boolean serve()
    { /* implementation not shown */ }
}

```

35. Which of the following declarations will cause an error? You may assume all the constructors are correctly implemented.

- (A) WeakPlayer t = new TennisPlayer("Smith");
- (B) TennisPlayer g = new GoodPlayer("Jones");
- (C) TennisPlayer w = new WeakPlayer("Henry");
- (D) TennisPlayer p = null;
- (E) WeakPlayer q = new WeakPlayer("Grady");

36. Refer to the `serve` method in the `WeakPlayer` class.

```
/** Returns true if serve is in (45% probability),
 *  false if serve is out (55% probability).
 */
public boolean serve()
{ /* implementation */ }
```

Which of the following replacements for `/* implementation */` satisfy the postcondition of the `serve` method?

- I. double value = Math.random();  
return value >= 0 || value < 0.45;
  - II. double value = Math.random();  
return value < 0.45;
  - III. int val = (int) (Math.random() \* 100);  
return val < 45;
- (A) I only
  - (B) II only
  - (C) III only
  - (D) II and III only
  - (E) I, II, and III

37. Consider the following class definition.

```
public class Beginner extends WeakPlayer
{
    private double costOfLessons;

    //methods of Beginner class
    ...
}
```

Refer to the following declarations and method in a client program.

```
TennisPlayer w = new WeakPlayer("Harry");
TennisPlayer b = new Beginner("Dick");
Beginner bp = new Beginner("Ted");

public void giveEncouragement(WeakPlayer t)
{ /* implementation not shown */ }
```

Which of the following method calls will cause an error?

- I. giveEncouragement(w);
  - II. giveEncouragement(b);
  - III. giveEncouragement(bp);
- (A) I only  
(B) II only  
(C) III only  
(D) I and II only  
(E) I, II, and III

38. A matrix class that manipulates matrices contains the following declaration.

```
private int[][] mat = new int[numRows][numCols];
```

Consider the following method that alters matrix `mat`.

```
public void doSomething()
{
    int width = mat[0].length;
    int numRows = mat.length;
    for (int row = 0; row < numRows; row++)
        for (int col = 0; col < width/2; col++)
            mat[row][col] = mat[row][width - 1 - col];
}
```

If `mat` has current value

```
1 2 3 4 5 6
1 3 5 7 9 11
```

what will the value of `mat` be after a call to `doSomething`?

- (A) 1 2 3 3 2 1  
1 3 5 5 3 1
- (B) 6 5 4 4 5 6  
11 9 7 7 9 11
- (C) 6 5 4 3 2 1  
11 9 7 5 3 1
- (D) 1 2 3 4 5 6  
1 2 3 4 5 6
- (E) 1 3 5 7 9 11  
1 3 5 7 9 11

Questions 39 and 40 refer to the following information.

Consider an array arr that is sorted in increasing order, and method findMost given below. Method findMost is intended to find the value in the array that occurs most often. If every value occurs exactly once, findMost should return -1. If there is more than one value that occurs the most, findMost should return any one of those. For example, if arr contains the values [1,5,7,7,10], findMost should return 7. If arr contains [2,2,2,7,8,8,8,9,9,9], findMost should return 2 or 9. If arr contains [1,2,7,8], findMost should return -1.

```

Line 1: /** Precondition: arr sorted in increasing order.
Line 2:  */
Line 3: public static int findMost(int[] arr)
Line 4: {
Line 5:     int index = 0;
Line 6:     int count = 1;
Line 7:     int maxCountSoFar = 1;
Line 8:     int mostSoFar = arr[0];
Line 9:     while (index < arr.length - 1)
Line 10:    {
Line 11:        while (index < arr.length - 1 &&
Line 12:                arr[index] == arr[index + 1])
Line 13:        {
Line 14:            count++;
Line 15:            index++;
Line 16:        }
Line 17:        if (count > maxCountSoFar)
Line 18:        {
Line 19:            maxCountSoFar = count;
Line 20:            mostSoFar = arr[index];
Line 21:        }
Line 22:        index++;
Line 23:    }
Line 24:    if (maxCountSoFar == 1)
Line 25:        return -1;
Line 26:    else
Line 27:        return mostSoFar;
Line 28: }
```

39. The method findMost does not always work as intended. An *incorrect* result will be returned if arr contains the values
- [1,2,3,4,5]
  - [6,6,6,6]
  - [1,2,2,3,4,5]
  - [1,1,3,4,5,5,5,7]
  - [2,2,2,4,5,5]
40. Which of the following changes should be made so that method findMost will work as intended?
- Insert the statement count = 1; between Lines 20 and 21.
  - Insert the statement count = 1; between Lines 21 and 22.
  - Insert the statement count = 1; between Lines 16 and 17.
  - Insert the statement count = 0; between Lines 23 and 24.
  - Insert the statement count = 1; between Lines 23 and 24.

# COMPUTER SCIENCE A

## SECTION II

Time—1 hour and 30 minutes

4 Questions

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

Write your answers in the separate Free-Response booklet provided.

**NOTES:**

- Assume that the classes in the Quick Reference have been imported where appropriate.
- 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.

1. A WordSet, whose partial implementation is shown in the class declaration below, stores a set of String objects in no particular order and contains no duplicates. Each word is a sequence of capital letters only.

```

public class WordSet
{
    /** Constructor initializes set to empty. */
    public WordSet()
    { /* implementation not shown */ }

    /** Returns the number of words in set. */
    public int size()
    { /* implementation not shown */ }

    /** Adds word to set.
     */
    public void insert(String word)
    { /* implementation not shown */ }

    /** Removes word from set if present, else does nothing.
     */
    public void remove(String word)
    { /* implementation not shown */ }

    /** Returns kth word in alphabetical order, where 1 <= k <= size().
     */
    public String findkth(int k)
    { /* implementation not shown */ }

    /** Returns true if set contains word, false otherwise. */
    public boolean contains(String word)
    { /* implementation not shown */ }

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

```

The `findkth` method returns the  $k$ th word in alphabetical order in the set, even though the implementation of `WordSet` may not be sorted. The number  $k$  ranges from 1 (corresponding to first in alphabetical order) to  $N$ , where  $N$  is the number of words in the set. For example, if `WordSet s` stores the words {"GRAPE", "PEAR", "FIG", "APPLE"}, here are the values when `s.findkth(k)` is called.

<code>k</code>	values of <code>s.findkth(k)</code>
1	APPLE
2	FIG
3	GRAPE
4	PEAR

Class information for this question

```
public class WordSet {
    public WordSet() { }
    public int size() { }
    public void insert(String word) { }
    public void remove(String word) { }
    public String findkth(int k) { }
    public boolean contains(String word) { }
}
```

- (a) Write a client method `countA` that returns the number of words in `WordSet s` that begin with the letter "A." In writing `countA`, you may call any of the methods of the `WordSet` class. Assume that the methods work as specified.

Complete method `countA`.

```
/** Returns the number of words in s that begin with "A".
 */
public static int countA(WordSet s)
```

- (b) Write a client method `removeA` that removes all words that begin with "A" from a non-null `WordSet`. If there are no such words in `s`, then `removeA` does nothing. In writing `removeA`, you may call method `countA` specified in part (a). Assume that `countA` works as specified, regardless of what you wrote in part (a).

Complete method `removeA`.

```
/** Removes from WordSet s all words that begin with the letter "A".
 * Precondition: WordSet is not null.
 * Postcondition: WordSet s contains no words that begin with
 * "A", but is otherwise unchanged.
 */
public static void removeA(WordSet s)
```

2. A clothing store sells shoes, pants, and tops. The store also allows a customer to buy an "outfit," which consists of three items: one pair of shoes, one pair of pants, and one top.

Each clothing item has a description and a price. The four types of clothing items are represented by the four classes Shoes, Pants, Top, and Outfit. All four classes are subclasses of a ClothingItem class, shown below.

```
public class ClothingItem
{
    private String description;
    private double price;

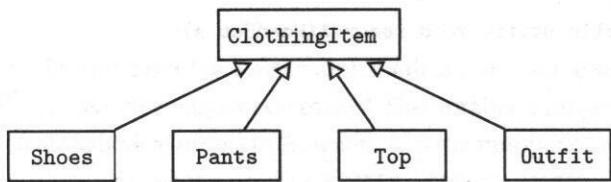
    public ClothingItem()
    {
        description = "";
        price = 0;
    }

    public ClothingItem(String descr, double aPrice)
    {
        description = descr;
        price = aPrice;
    }

    public String getDescription()
    {
        return description;
    }

    public double getPrice()
    {
        return price;
    }
}
```

The following diagram shows the relationship between the ClothingItem class and the Shoes, Pants, Top, and Outfit classes.



The store allows customers to create Outfit clothing items, each of which includes a pair of shoes, pants, and a top. The description of the outfit consists of the description of the shoes, pants, and top, in that order, separated by "/" and followed by a space and "outfit". The price of an outfit is calculated as follows. If the sum of the prices of any two items equals or exceeds \$100, there is a 25% discount on the sum of the prices of all three items. Otherwise there is a 10% discount.

For example, an outfit consisting of sneakers (\$40), blue jeans (\$50), and a T-shirt (\$10) would have the name "sneakers/blue jeans/T-shirt outfit" and a price of  $0.90(40 + 50 + 10) = \$90.00$ . An outfit consisting of loafers (\$50), cutoffs (\$20), and

dress-shirt (\$60) would have the description "loafers/cutoffs/dress-shirt outfit" and price  $0.75(50 + 20 + 60) = \$97.50$ .

Write the `Outfit` subclass of `ClothingItem`. Your implementation must have just one constructor that takes three parameters representing a pair of shoes, pants, and a top, in that order.

A client class that uses the `Outfit` class should be able to create an outfit, get its description, and get its price. Your implementation should be such that the client code has the following behavior:

```

Shoes shoes;
Pants pants;
Top top;
/* Code to initialize shoes, pants, and top */

ClothingItem outfit =
    new Outfit (shoes, pants, top); //Compiles without error
ClothingItem outfit =
    new Outfit (pants, shoes, top); //Compile-time error
ClothingItem outfit =
    new Outfit (shoes, top, pants); //Compile-time error

```

3. Consider a note keeper object that is designed to store and manipulate a list of short notes. Here are some typical notes:

```

pick up drycleaning
special dog chow
car registration
dentist Monday
dog license

```

A note is represented by the following class.

```

public class Note
{
    /** Returns a one-line note. */
    public String getNote()
    { /* implementation not shown */ }

    //There may be instance variables, constructors, and methods
    //that are not shown.
}

```

A note keeper is represented by the NoteKeeper class shown below.

```
public class NoteKeeper
{
    /** The list of notes */
    private ArrayList<Note> noteList;

    /** Prints all notes in noteList, as described in part(a).
     */
    public void printNotes()
    { /* to be implemented in part (a) */

        /** Removes all notes with specified string from noteList,
         * as described in part (b).
         * If none of the notes in noteList contains the given string,
         * the list remains unchanged.
         */
        public void removeNotes(String str)
        { /* to be implemented in part (b) */

            //There may be instance variables, constructors, and methods
            //that are not shown.
        }
    }
}
```

- (a) Write the NoteKeeper method `printNotes`. This method prints all of the notes in `noteList`, one per line, and numbers the notes, starting at 1. The output should look like this.

1. pick up drycleaning
2. special dog chow
3. car registration
4. dentist Monday
5. dog license

Complete method `printNotes`.

```
/** Prints all notes in noteList, as described in part(a).
 */
public void printNotes()
```

- (b) Write the NoteKeeper method `removeNotes`. Method `removeNotes` removes all notes from `noteList` that contain the string specified by the parameter. The ordering of the remaining notes should be left unchanged. For example, suppose that a `NoteKeeper` variable, `notes`, has a `noteList` containing the following.

```
[pick up drycleaning, special dog chow, car registration,  
dentist Monday, dog license]
```

The method call `notes.removeNotes("dog")` should modify the `noteList` of `notes` to be

```
[pick up drycleaning, car registration, dentist Monday]
```

The method call `notes.removeNotes("cow")` should leave the list shown above unchanged.

Here's another example. If `noteList` contains

```
[pick up car, buy carrots, dog license, carpet cleaning]
```

the method call `notes.removeNotes("car")` should modify the `noteList` of `notes` to be

```
[dog license]
```

Complete method `removeNotes`.

```
/** Removes all notes with specified string from noteList,  
 * as described in part (b).  
 * If none of the notes in noteList contains the given string,  
 * the list remains unchanged.  
 */  
public void removeNotes(String str)
```

4. Consider the problem of keeping track of the available seats in a theater. Theater seats can be represented with a two-dimensional array of integers, where a value of 0 shows a seat is available, while a value of 1 indicates that the seat is occupied. For example, the array below shows the current seat availability for a show in a small theater.

	[0]	[1]	[2]	[3]	[4]	[5]
[0]	0	0	1	1	0	1
[1]	0	1	0	1	0	1
[2]	1	0	0	0	0	0

The seat at slot [1][3] is taken, but seat [0][4] is still available.

A show can be represented by the Show class shown below.

```
public class Show
{
    /** The seats for this show */
    private int[][] seats;

    private final int SEATS_PER_ROW = <some integer value>;
    private final int NUM_ROWS = <some integer value>;

    /** Reserve two adjacent seats and return true if this was
     * successfully done, false otherwise, as described in part (a).
     */
    public boolean twoTogether()
    { /* to be implemented in part (a) */ }

    /** Return the lowest seat number in the specified row for a
     * block of seatsNeeded empty adjacent seats, as described in part (b).
     */
    public int findAdjacent(int row, int seatsNeeded)
    { /* to be implemented in part (b) */ }

    //There may be instance variables, constructors, and methods
    //that are not shown.
}
```

- (a) Write the Show method `twoTogether`, which reserves two adjacent seats and returns `true` if this was successfully done. If it is not possible to find two adjacent seats that are unoccupied, the method should leave the show unchanged and return `false`. For example, suppose this is the state of a show.

	[0]	[1]	[2]	[3]	[4]	[5]
[0]	0	0	1	1	0	1
[1]	0	1	0	1	0	1
[2]	1	0	0	0	1	1

A call to `twoTogether` should return `true`, and the final state of the show could be any one of the following three configurations.

	[0]	[1]	[2]	[3]	[4]	[5]
[0]	1	1	1	1	0	1
[1]	0	1	0	1	0	1
[2]	1	0	0	0	1	1

OR

	[0]	[1]	[2]	[3]	[4]	[5]
[0]	0	0	1	1	0	1
[1]	0	1	0	1	0	1
[2]	1	1	1	0	1	1

OR

	[0]	[1]	[2]	[3]	[4]	[5]
[0]	0	0	1	1	0	1
[1]	0	1	0	1	0	1
[2]	1	0	1	1	1	1

For the following state of a show, a call to `twoTogether` should return `false` and leave the two-dimensional array as shown.

	[0]	[1]	[2]	[3]	[4]	[5]
[0]	0	1	0	1	1	0
[1]	1	1	0	1	0	1
[2]	0	1	1	1	1	1

Class information for this question

```
public class Show {
    private int[][] seats;
    private final int SEATS_PER_ROW;
    private final int NUM_ROWS;
    public boolean twoTogether();
    public int findAdjacent(int row, int seatsNeeded);
```

Complete method `twoTogether`.

```
/** Reserve two adjacent seats and return true if this was
 * successfully done, false otherwise, as described in part (a).
 */
public boolean twoTogether()
```

- (b) Write the `Show` method `findAdjacent`, which finds the lowest seat number in the specified row for a specified number of empty adjacent seats. If no such block of empty seats exists, the `findAdjacent` method should return `-1`. No changes should be made to the state of the show, irrespective of the value returned.

For example, suppose the diagram of seats is as shown.

	[0]	[1]	[2]	[3]	[4]	[5]
[0]	0	1	1	0	0	0
[1]	0	0	0	0	1	1
[2]	1	0	0	1	0	0

The following table shows some examples of calling `findAdjacent` for `Show`.

Method call	Return value
<code>show.findAdjacent(0, 3)</code>	3
<code>show.findAdjacent(1, 3)</code>	0 or 1
<code>show.findAdjacent(2, 2)</code>	1 or 4
<code>show.findAdjacent(1, 5)</code>	-1

Complete method `findAdjacent`.

```
/** Return the lowest seat number in the specified row for a
 * block of seatsNeeded empty adjacent seats, as described in part (b).
 */
public int findAdjacent(int row, int seatsNeeded)
```

**STOP**

**END OF EXAM**