Practice Test 2

AP® Computer Science A Exam

SECTION I: Multiple-Choice Questions

DO NOT OPEN THIS BOOKLET UNTIL YOU ARE TOLD TO DO SO.

At a Glance

Total Time
1 hour 30 minutes
Number of Questions
40
Percent of Total Score
50%
Writing Instrument
Pencil required

Instructions

Section I of this examination contains 40 multiple-choice questions. Fill in only the ovals for numbers 1 through 40 on your answer sheet.

Indicate all of your answers to the multiple-choice questions on the answer sheet. No credit will be given for anything written in this exam booklet, but you may use the booklet for notes or scratch work. After you have decided which of the suggested answers is best, completely fill in the corresponding oval on the answer sheet. Give only one answer to each question. If you change an answer, be sure that the previous mark is erased completely. Here is a sample question and answer.

Sample Question	Sample Answer
Chicago is a (A) state (B) city (C) country (D) continent (E) county	

Use your time effectively, working as quickly as you can without losing accuracy. Do not spend too much time on any one question. Go on to other questions and come back to the ones you have not answered if you have time. It is not expected that everyone will know the answers to all the multiple-choice questions.

About Guessing

Many candidates wonder whether or not to guess the answers to questions about which they are not certain. Multiple-choice scores are based on the number of questions answered correctly. Points are not deducted for incorrect answers, and no points are awarded for unanswered questions. Because points are not deducted for incorrect answers, you are encouraged to answer all multiple-choice questions. On any questions you do not know the answer to, you should eliminate as many choices as you can, and then select the best answer among the remaining choices.

GO ON TO THE NEXT PAGE.

Java Quick Reference

Class Constructors and Methods	Explanation
String Class	
`String(String str)`	Constructs a new `String` object that represents the same sequence of characters as `str`
`int length()`	Returns the number of characters in a `String` object
`String substring(int from, int to)`	Returns the substring beginning at index `from` and ending at index `to - 1`
`String substring(int from)`	Returns `substring(from, length())`
`int indexOf(String str)`	Returns the index of the first occurrence of `str`; returns `-1` if not found
`boolean equals(String other)`	Returns `true` if `this` is equal to `other`; returns `false` otherwise
`int compareTo(String other)`	Returns a value <0 if `this` is less than `other`; returns zero if `this` is equal to `other`; returns a value of >0 if `this` is greater than `other`
Integer Class	
`Integer(int value)`	Constructs a new `Integer` object that represents the specified `int` value
`Integer.MIN_VALUE`	The minimum value represented by an `int` or `Integer`
`Integer.MAX_VALUE`	The maximum value represented by an `int` or `Integer`
`int intValue()`	Returns the value of this `Integer` as an `int`
Double Class	
`Double(double value)`	Constructs a new `Double` object that represents the specified `double` value
`double doubleValue()`	Returns the value of this `Double` as a `double`
Math Class	
`static int abs(int x)`	Returns the absolute value of an `int` value
`static double abs(double x)`	Returns the absolute value of a `double` value
`static double pow(double base, double exponent)`	Returns the value of the first parameter raised to the power of the second parameter
`static double sqrt(double x)`	Returns the positive square root of a `double` value
`static double random()`	Returns a `double` value greater than or equal to `0.0` and less than `1.0`
ArrayList Class	
`int size()`	Returns the number of elements in the list
`boolean add(E obj)`	Appends `obj` to end of list; returns `true`
`void add(int index, E obj)`	Inserts `obj` at position index (`0 <= index <= size`), moving elements at position `index` and higher to the right (adds `1` to their indices) and adds 1 to size
`E get(int index)`	Returns the element at position `index` in the list
`E set(int index, E obj)`	Replaces the element at position `index` with `obj`; returns the element formerly at position `index`
`E remove(int index)`	Removes element from position `index`, moving elements at position `index + 1` and higher to the left (subtracts `1` from their indices) and subtracts 1 from size; returns the element formerly at position `index`
Object Class	
`boolean equals(Object other)`	
`String toString()`	

GO ON TO THE NEXT PAGE.

COMPUTER SCIENCE A

SECTION I

Time—1 hour and 30 minutes

Number of Questions—40

Percent of total exam 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. No credit will be given for anything written in the examination booklet. Do not spend too much time on any one problem.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Assume that declarations of variables and methods appear within the context of an enclosing class.
- Assume that method calls that are not prefixed with an object or class name and are not shown within a complete class definition appear within the context of an enclosing class.
- Unless otherwise noted in the question, assume that parameters in the method calls are not `null` and that methods are called only when their preconditions are satisfied.

1. Consider the following method.

```
public static int mystery(int a, int b)
{
   if (a <= 0)
        return b;
   else
        return mystery(a - 2, b);
}
```

What value is returned by the call `mystery(12, 5)`?

(A) `5`
(B) `6`
(C) `12`
(D) `60`
(E) `1565`

GO ON TO THE NEXT PAGE.

2. Consider the following instance variable and method.

```
private int[] numList;

//Precondition: numList contains a list of int values in no particular order

public int mystery(int n)
{
    for (int k = 0; k <= numList.length - 1; k++)
    {
       if (n <= numList[k])
          return k;
    }
    return numList.length;
}
```

Which of the following statements is most accurate about `numList` following the execution of the following statement?

```
int j = mystery(number);
```

(A) The greatest value in `numList` is at index `j`.
(B) The greatest value in `numList` that is less than `number` is at index `j`.
(C) All values in `numList` from index `0` to `j - 1` are greater than or equal to `number`.
(D) All values in `numList` from index `0` to `j - 1` are less than `number`.
(E) All values in `numList` from index `j` to `numList.length - 1` are greater than `number`.

GO ON TO THE NEXT PAGE.

Questions 3–4 refer to the following incomplete class declaration for a new data type called a `Quack`.

```
Public class Quack
{
   ArrayList<Object> myData;

   // Constructor initializes myData
   public Quack()
   {  /* implementation not shown */  }

   // Quack.
   public void enquack(Object x)
   {  /* implementation not shown */  }

   // if front is true, returns the object at the front end of the Quack; otherwise returns the object at the back end of the
   // Quack.  Assumes the Quack is not empty.
   public Object dequack(boolean front)
   {  /* implementation not shown */  }

   // Returns true if the Quack has no objects; otherwise returns false.
   public boolean isEmpty()
   {  /* implementation not shown */  }

   <designation> ArrayList myData;

   // ... other methods and data not shown
```

3. Which of the following is the best choice for `<designation>` and the best reason for that choice?

   (A) `<designation>` should be `private` so that programs using a `Quack` will not be able to modify `myData` by using methods `enquack` and `dequack`, thereby preserving the principle of data stability.

   (B) `<designation>` should be `private` so that programs using a `Quack` can modify `myData` only by using methods such as `enquack` and `dequack`, thereby preserving the principle of information hiding.

   (C) `<designation>` should be `private` as an indication to programs using a `Quack` that `myData` can be modified directly but that it is *better* to modify `myData` only by using methods such as `enquack` and `dequack`, thereby preserving the principle of maximum information dissemination.

   (D) `<designation>` should be `public` because programs using a `Quack` need to know how the `Quack` class has been implemented in order to use it.

   (E) `<designation>` should be `public`. Otherwise, only objects constructed from derived subclasses of a `Quack` will be able to modify the contents of a `Quack`.

4. Which of the following is an effective return statement for `isEmpty` as described in the incomplete declaration above?

   (A) `return (myData.length == 0)`
   (B) `return (size() == 0)`
   (C) `return (myData.size() == 0);`
   (D) `return (myData.length() == 0)`
   (E) `return (myData.size == 0)`

GO ON TO THE NEXT PAGE.

5. Consider the following method definition.

```
public static int mystery(int n)
{
   if (n <= 1)
      return 2;
   else
      return 1 + mystery(n - 3);
}
```

Which of the following lines of code can replace the line in `mystery` containing the recursive call so that the functionality of `mystery` does not change?

(A) `return 1 + ((n + 2) / 3);`
(B) `return 1 + ((n + 3) / 2);`
(C) `return 2 + ((n + 1) / 3);`
(D) `return 2 + ((n + 2) / 3);`
(E) `return 3 + ((n + 1) / 2);`

GO ON TO THE NEXT PAGE.

Questions 6–7 refer to the following incomplete class declaration.

```
public class DistanceTracker
{
    private int kilometers;
    private int meters;

    /* Constructs a DistanceTracker object
     * @param k the number of kilometers
     * Precondition: k ≥ 0
     * @param m the number of meters
     * Precondition: 0 ≤ m < 1000
     */
     public DistanceTracker (int k, int m)
     {
       kilometer = k;
       meters = m;
     }
     /* @return the number of kilometers
     */
     public int getKilometers()
     { /* implementation not shown */}
     /* @return the number of meters
     */
     public int getMeters()
     { /* implementation not shown */}
       /* Adds k kilometers and m meters
        * @param k the number of kilometers
        * Precondition: k ≥ 0
        *@param m the number of meters
     * Precondition: m ≥ 0
     */
     public void addDistance(int k, int m)
     {
       kilometers += k;
       meters += m;
       /* rest of method not shown */
     }
    //Rest of class not shown
    }
```

6. Which of the following code segments can be used to replace `/* rest of method not shown */` so `addDistance` will correctly increase the distance?

(A)
```
kilometers += meters / 1000
meters = meters % 1000
```

(B)
```
kilometers += meters % 1000
meters = meters / 1000
```

(C)
```
meters += kilometers % 1000
```

(D)
```
kilometers += meters % 1000
```

(E)
```
meters = meters % 1000
```

GO ON TO THE NEXT PAGE.

7. Consider the following incomplete class declaration.

```
public class DistanceTrackerSet
{
    Distance Tracker[] set;
    /*Declaration method not shown*/

    public  DistanceTracker total()
    {
       DistanceTracker temp = new DistanceTracker(0, 0);
    for (int k = 0; k < set.length; k++)
       {
          /* missing code */
       }
       return temp;
    }
/*Other methods not shown*/
}
```

Assuming `set` is properly initialized with `DistanceTracker` objects and all needed classes are properly imported, which of the following can be used to replace `/* missing code */` so that the method returns a `DistanceTracker` object with the total of all distances stored in `set`?

(A) `temp.addDistance(temp[k].kilometers, temp[k].meters);`
(B) `set[k].addDistance(temp[k].getKilometers(), temp[k].getMeters());`
(C) `set[k].addDistance();`
(D) `temp += temp.addDistance();`
(E) `temp.addDistance(temp[k].getKilometers(), temp[k].getMeters());`

8. Consider the following method.

```
public List<Integer> nums()
{
   List<Integer> values = new ArrayList<Integer>();
   for (int i = 0; i < 50; i = i + 5)
      if (i % 4 == 1)
         values.add(i);
   return value;
}
```

What will the return of `nums()` contain?

(A) `[5, 45]`
(B) `[5, 25, 45]`
(C) `[0, 20, 40]`
(D) `[5, 9, 13, 17, 21, 25, 29, 33, 37, 41, 45]`
(E) `[0, 5, 10, 15, 20, 25, 30, 35, 40, 45]`

GO ON TO THE NEXT PAGE.

9. Consider the following incomplete method `mystery`.

```
public static boolean mystery(boolean a, boolean b, boolean c)
{
   return <expression>;
}
```

What should `<expression>` be replaced with so that `mystery` returns `true` when exactly two of its three parameters are `true`; otherwise `mystery` returns `false`?

(A)
```
(a && b && !c) ||
(a && !b && c) ||
(!a && b && c)
```
(B)
```
(a && b && !c) &&
(a && !b && c) &&
(!a && b && c)
```
(C)
```
(a || b || !c) &&
(a || !b || c) &&
(!a || b || c)
```
(D)
```
(a && b) ||
(a && c) ||
(b && c)
```
(E)
```
(a || b) &&
(a || c) &&
(b || c)
```

10. Consider the following code segment.

```
int x;
x = 5 - 4 + 9 * 12 / 3 - 10;
```

What is the value of `x` after the code segment is executed?

(A) `13`
(B) `27`
(C) `30`
(D) `-57`
(E) `-10`

11. What is the best way to declare a variable `myStrings` that will store 50 `String` values if each `String` will be no longer than 25 characters?

(A) `ArrayList <String> myStrings[String[50]];`
(B) `ArrayList <String> myStrings = new String[50];`
(C) `ArrayList <String> myStrings = new String[25];`
(D) `String [] myStrings = new String [50, 25];`
(E) `String [] myStrings = new String [50];`

GO ON TO THE NEXT PAGE.

12. Consider the following code segment.

```
List <Integer> scores = new ArrayList<Integer>();
scores.add(93);
scores.add(97);
scores.add(84);
scores.add(91);
scores.remove(2);
scores.add(1, 83);
scores.set(3, 99);
System.out.println(scores);
```

What is the output of the code segment?

(A) `[83, 93, 99, 91]`
(B) `[93, 83, 91, 99]`
(C) `[83, 94, 91, 99]`
(D) `[93, 83, 97, 99]`
(E) The code throws an `ArrayIndexOutofBoundsException`.

13. Consider the following precondition, postcondition, and signature for the `getDigit` method.

```
// precondition: n >= 0
//               whichDigit >= 0
// postcondition: Returns the digit of n in
//                the whichDigit position
//                when the digits of n are
//                numbered from right to
//                left starting with zero.
//                Returns 0 if whichDigit  >=
//                number of digits of n.
int getDigit (int n, int whichDigit)
```

Consider also the following three possible implementations of the `getDigit` method.

I.
```
if (whichDigit == 0)
  return n % 10;
else
  return getDigit(n / 10, whichDigit - 1) ;
```

II.
```
return (n / (int) Math.pow(10, whichDigit) )  % 10;
```

III.
```
for (int k = 0; k < whichDigit; k++)
  n /= 10;
return n % 10;
```

Which implementation(s) would satisfy the postcondition of the `getDigit` method?

(A) I and II only
(B) I and III only
(C) II and III only
(D) I, II, and III
(E) None of the above

GO ON TO THE NEXT PAGE.

14. Consider an array of integers.

```
4    10    1    2    6    7    3    5
```

Assume that `SelectionSort` is used to order the array from smallest to largest values.

Which of the following represents the state of the array immediately after the first iteration of the outer `for` loop in the `SelectionSort` process?

(A) `1 4 10 2 3 6 7 5`
(B) `1 2 4 6 10 7 3 5`
(C) `1 10 4 2 6 7 3 5`
(D) `4 3 1 5 6 7 10 2`
(E) `5 3 7 6 2 1 10 4`

15. Assume that a program declares and initializes `v` as follows.

```
String [] v;
v = initialize () ;          // Returns an array of
                             // length 10 containing
                             // ten valid strings
```

Which of the following code segments correctly traverses the array *backward* and prints out the elements (one per line)?

I.
```
for (int k = 9; k >= 0; k--)
    System.out.println(v[k]) ;
```

II.
```
int k = 0;
while (k < 10)
{
  System.out.println(v[9 - k]);
  k++;
}
```

III.
```
int k = 10;
while (k >= 0)
{
  System.out.println(v[k]);
  k-- ;
 }
```

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

GO ON TO THE NEXT PAGE.

16. Consider the following method.

```
/**Precondition: @param set an ArrayList that contains distinct integers
*@param n an int value
*/
public int mystery(List<Integer> set, int n)
{
    for (int k = 0; k < set.length(); k++)
    {
       if (set.get(k) > n)
       {
          return (set.remove(k) + mystery(set, n));
       }
    }
    return 0;
}
```

What is returned by the method call `mystery(set, n)`?

(A) 0
(B) The number of elements of `set` that are greater than `n`
(C) The sum of the elements of `set` that are greater than `n`
(D) The sum of the elements of `set` that are less than `n`
(E) The sum of the elements of `set` that are less than or equal to `n`

GO ON TO THE NEXT PAGE.

17. Consider the following two-dimensional array.

```
[ [0, 0, 0 ,0]
  [0, 1, 0, 0]
  [0, 1, 2, 0]
  [0, 1, 2, 3] ]
```

Which of the following methods returns this two-dimensional array?

(A)
```
public int[][] nums()
{
  int[][] temp = new int[4][4];
  for (int j = 0; j < 4; j++)
  {
      for (int k = 0; k < 4; k++)
      {
            temp[j][k] = j;
      }
  }
  return temp;
}
```

(B)
```
public int[][] nums()
{
  int[][] temp = new int[4][4];
  for (int j = 0; j < 4; j++)
  {
      for (int k = 0; k < 4; k++)
      {
            temp[j][k] = k;
      }
  }
  return temp;
}
```

(C)
```
public int[][] nums()
{
  int[][] temp = new int[4][4];
  for (int j = 0; j < 4; j++)
  {
      for (int k = j; k < 4; k++)
      {
            temp[j][k] = k;
      }
  }
  return temp;
}
```

(D)
```
public int[][] nums()
{
  int[][] temp = new int[4][4];
  for (int j = 0; j < 4; j++)
  {
      for (int k = 0; k <= j; k++)
      {
            temp[j][k] = j;
      }
  }
  return temp;
}
```

(E)
```
public int[][] nums()
{
  int[][] temp = new int[4][4];
  for (int j = 0; j < 4; j++)
  {
      for (int k = 0; k <= j; k++)
      {
            temp[j][k] = k;
      }
  }
  return temp;
}
```

GO ON TO THE NEXT PAGE.

18. A children's club classifies members based on age according to the table below:

Years	Classification
Under 3	Infant
3 to 7 inclusive	Pee-Wee
8 to 13 inclusive	Cub
Over 14	Leader

Which of the following methods will correctly take the integer parameter `age` and return the string `Classification`?

(A)
```
public String Classification(int age)
{
   String temp;
   if (age < 3)
      temp = "Infant";
   if (age <= 7)
      temp = "Pee-Wee";
   if (age <= 13)
      temp = "Cub";
   if (age >= 14)
      temp = "Leader";
   return temp;
}
```

(B)
```
public String Classification(int age)
{
   String temp;
   if (age < 3)
      temp = "Infant";
   if (3 <= age <= 7)
      temp = "Pee-Wee";
   if (8 <= age <= 13)
      temp = "Cub";
   if (age >= 14)
      temp = "Leader";
   return temp;
}
```

(C)
```
public String Classification(int age)
{
   String temp;
   if (age < 3)
      temp = "Infant";
   else if (age <= 7)
      temp = "Pee-Wee";
   else if (age <= 13)
      temp = "Cub";
   else if (age > 14)
      temp = "Leader";
   return temp;
}
```

(D)
```
public String Classification(int age)
{
   String temp;
   if (age < 3)
      temp = "Infant";
   else if (age < 7)
      temp = "Pee-Wee";
   else if (age < 13)
      temp = "Cub";
   else if (age > 14)
      temp = "Leader";
   return temp;
}
```

(E)
```
public String Classification(int age)
{
   String temp;
   if (age < 3)
      temp = "Infant";
   if (age < 7)
      temp = "Pee-Wee";
   if (age < 13)
      temp = "Cub";
   if (age > 14)
      temp = "Leader";
   return temp;
}
```

GO ON TO THE NEXT PAGE.

Questions 19–20 refer to the method `getGap` with line numbers added for reference. Method `getGap` is intended to find the maximum difference between the indices of any two occurrences of `num` in the array `arr`. However, the method `getGap` does not work as intended.

For example, if the array `arr` contains [8, 7, 5, 5, 4, 7, 2, 7, 1, 2, 7], the call `getGap(arr, 7)` should return 9, the difference between the indices of the first and last occurrences of 7.

/**Precondition: `arr` contains at least two occurrences of `num` */

```
1     public int getGap(int[] arr, int num)
2     {
3          int index1 = -1;
4          int index2 = -1;
5          for (int k = 0; k < arr.length; k++)
6          {
7               if (arr[k] == num)
8               {
9                    if (index1 == -1)
10                   {
11                        index1 = k;
12                        index2 = k;
13                   }
14                   else
15                   {
16                        index1 = index2;
17                        index2 = k;
18                   }
19              }
20         }
21         return (index2 - index1);
22    }
```

GO ON TO THE NEXT PAGE.

19. The method `getGap` does not work as intended. Which of the following best describes the return of the method `getGap` ?

    (A) The difference between the indices of the last two occurrences of `num` in `arr`
    (B) The minimum difference between the indices of any two occurrences of `num` in `arr`
    (C) The difference between the indices of the first two occurrences of `num` in `arr`
    (D) The length of the array `arr`
    (E) The number of occurrences of `num` in `arr`

20. Which of the following changes should be made to `getGap` so that the method will work as intended?

    (A) Delete the statement at line 4.
    (B) Delete the statement at line 11.
    (C) Delete the statement at line 12.
    (D) Delete the statement at line 16.
    (E) Delete the statement at line 17.

GO ON TO THE NEXT PAGE.

Questions 21–23 refer to the following incomplete class declaration that is intended to be used to represent calendar dates.

```
Public class Date
{
   private int month;
       // represents month 0–11
   private int day;
       // represents day of the month
       // 0–31
   private int year;
       // represents the year

   // constructor sets the private data
   public Date (int m, int d, int y)
   { /* implementation not shown */ }

   // postconditions: returns the month
   public int getMonth()
   { /* implementation not shown */ }

   // postcondition: return the day
   public int getDay()
   { /* implementation not shown */ }

   // postcondition: returns the year
   public int getYear()
   { /* implementation not shown */ }

   // postcondition: returns the number of
   //                days which, when
   //                added to this Date,
   //                gives newDate
   public int daysUntil (Date newDate)
   { /* implementation not shown */ }

   // postcondition: returns true if
   //                the month, day, and
   //                year of this Date are
   //                are equal to those of
   //                other; otherwise
   //                returns false
   public boolean equals (Date other)
   { /* implementation not shown */ }

   // ... other methods not shown
```

GO ON TO THE NEXT PAGE.

21. Consider the method `equals` of the `Object` class.

    Which of the following method signatures is appropriate for the `equals` method?

    (A) `public boolean equals (Object other)`
    (B) `public int equals (Object other)`
    (C) `public boolean equals (Date other)`
    (D) `public int equals (Date other)`
    (E) `public boolean equals (Date d1, Date d2)`

22. Which of the following code segments could be used to implement the `equals` method of the `Date` class so that the `equals` method works as intended?

    I.
    ```
    if (month == other.month)
      if (day == other.day)
            if (year == other.year)
                  return true;
    Return false;
    ```
    II.
    ```
    if (month == other.getMonth() &&
            day == other.getDay() &&
            year == other.getYear())
      return true;
    else
      return false;
    ```
    III.
    ```
    return !((getMonth()  != other.getMonth())  ||
      (getDay()  != other.getDay())  ||
      (getYear() != other.getYear())) ;
    ```

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

23. During the testing of the `Date` class, it is determined that the class does not correctly handle leap years—although it handles non-leap years correctly.

    In which method of the `Date` class is the problem most likely to be found?

    (A) The `Date` constructor
    (B) The `getMonth` method
    (C) The `getDay` method
    (D) The `daysUntil` method
    (E) The `equals` method

GO ON TO THE NEXT PAGE.

24. Consider the following methods:

```
public static void mystery()
{
   int [] A;
   A = initialize ();
       // returns a valid initialized
       // array of integers
   for (int k = 0; k < A.length / 2; k++)
      swap (A[k], A[A.length - k - 1]);
}

public static void swap (int x, int y)
{
   int temp;
   temp = x;
   x = y;
   y = temp;
}
```

Which of the following best characterizes the effect of the `for` loop in the method `mystery`?

(A) It sorts the elements of `A`.
(B) It reverses the elements of `A`.
(C) It reverses the order of the first half of `A` and leaves the second half unchanged.
(D) It reverses the order of the second half of `A` and leaves the first half unchanged.
(E) It leaves all of the elements of `A` in their original order.

GO ON TO THE NEXT PAGE.

25. Consider the following code segment:

```
int [][] A = new int [4][3] ;
for (int j = 0; j < A[0].length; j++)
   for (int k = 0; k < A.length; k++)
      if (j == 0)
           A[k][j] = 0;
      else if (k % j == 0)
           A[k][j] = 1;
      else
           A[k][j] = 2;
```

What are the contents of `A` after the code segment has been executed?

(A)
```
0 0 0 0
1 1 1 1
1 2 1 2
```
(B)
```
0 1 1 1
0 2 2 2
0 1 2 1
```
(C)
```
0 0 0
1 1 2
1 1 1
1 1 2
```
(D)
```
0 1 1
0 2 1
0 2 2
0 2 1
```
(E)
```
0 1 1
0 1 2
0 1 1
0 1 2
```

26. Consider the following method:

```
/** @param num an int value such that num >= 0
 */
public void mystery(int num)
{
   System.out.print(num % 100);
   if ((num / 100) != 0)
   {
      mystery(num / 100);
   }
   System.out.print(num % 100);
}
```

Which of the following is printed as a result of the call `mystery(456789)`?

(A) `456789`
(B) `896745`
(C) `987654`
(D) `456789896745`
(E) `896745456789`

GO ON TO THE NEXT PAGE.

27. Consider the following method:

```
public static int mystery(int x, int y)
{
   if (x > 0)
      return x;
   else if (y > 0)
      return y;
   else
      return x / y;
}
```

In accordance with good design and testing practices, which of the following is the best set of test cases (`x, y`) for the method `mystery`?

(A) `(3, 4), (-3, 4), (-3, -4)`
(B) `(3, 4), (-3, 4), (-3, -4), (-3, 0)`
(C) `(3, 4), (3, -4), (-3, -4), (-3, 0)`
(D) `(3, 4), (3, -4), (3, 0), (-3, 4), (-3, -4)`
(E) `(3, 4), (2, 5), (3, -4), (-3, 0), (4, 0), (0, 0)`

28. Consider the following method.

```
/** Precondition: numList is not empty
 */
private int mystery(int[] numList)
{
   int n = numList[numList.length - 1];
   for (int k : numList)
   {
    if (n > k)
       {
        n = k;
       }
   }
   return n;
}
```

Which of the following best describes the return of `mystery`?

(A) The largest value in the array `numList`
(B) The least value in the array `numList`
(C) The index of the largest value in the array `numList`
(D) The index of the least value in the array `numList`
(E) The number of indices whose values are less than `numList[n]`

GO ON TO THE NEXT PAGE.

29. Consider the following method.

```
public int[] editArray(int[] arr, int oldNum, int newNum)
{
    /* missing code */
    return arr;
}
```

The method above is intended to replace any instance of `oldNum` in `arr` with any instances of `newNum`. Which of the following can be used to replace `/* missing code */` to replace any values of `oldNum` in the array with values of `newNum`?

(A)
```
for (int k = 0; k < arr.length; k++)
{
   if (arr[k] = oldNum)
   {
      arr[k] == newNum;
   }
}
```

(B)
```
for (int k = 0; k < arr.length; k++)
{
   if (arr[k] == oldNum)
   {
      arr[k] = newNum;
   }
}
```

(C)
```
while (arr[k] == oldNum)
{
   arr[k] = newNum
}
```

(D)
```
for (int k = 0; k < arr.length; k++)
{
   arr[k] == newNum;
}
```

(E)
```
while (int k = 0; k < arr.length; k++)
{
   if (arr[k] = oldNum)
   {
      arr[k] == newNum;
   }
}
```

GO ON TO THE NEXT PAGE.

30. Consider the following two classes.

```
public class SalesPerson
{
    public void sale()
    {
       System.out.print("greet ");
       pitch();
    }
    public void pitch()
    {
       System.out.print("pitch ");
    }

}

public class CommissionedSalesPerson extends SalesPerson
{
    public void sale()
    {
       super.sale();
       System.out.print("record ");
    }
    public void pitch()
    {
       super.pitch();
       system.out.print("close ");
    }

}
```

The following code segment is found in a class other than `SalesPerson`.

```
SalesPerson vincent = new CommissionedSalesPerson();
vincent.sale();
```

Which of the following is the best description of the functionality of this code segment?

(A) `greet pitch`
(B) `greet pitch close`
(C) `greet pitch record`
(D) `greet pitch record close`
(E) `greet pitch close record`

GO ON TO THE NEXT PAGE.

31. Consider the following declaration of a class that will be used to represent dimensions of rectangular crates.

```
public class Crate
{
    private int length;
    private int width;
    private int height;

    public Crate(int x, int y, int z)
    {
       length = x;
       width = y;
       height = z;
    }

    //other methods not shown
}
```

The following incomplete class declaration is intended to extend the `Crate` class so that the color of the crate can be specified.

```
public class ColoredCrate extends Crate
{
   private String color;
   //Constructors not shown
   //Other methods not shown
}
```

Which of the following possible constructors for `ColoredCrate` would be considered legal?

I.
```
public ColoredCrate(int a, int b, int c, String crateColor)
{
   length = a;
   width = b;
   height = c;
   color = crateColor;
}
```

II.
```
public ColoredCrate(int a, int b, int c, String crateColor)
{
   super (a, b, c);
   color = crateColor;
}
```

III.
```
public ColoredCrate()
{
   color = "";
}
```

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

GO ON TO THE NEXT PAGE.

32. Consider the following three proposed implementations of method `reverse`, intended to return an `ArrayList` of the objects in reversed order:

I.
```
public static ArrayList<Object> reverse (ArrayList<Object> q)
{
      ArrayList<Object> s = new ArrayList<Object>();
      while (q.size() != 0)
            s.add(0, q.remove(0));
      return s;
}
```

II.
```
public static ArrayList<Object> reverse (ArrayList<Object> q)
{
      ArrayList<Object> s = new ArrayList<Object>();
      for (int k = 0; k < q.size(); k++)
            s.add(0, q.remove(0));
      return s;
}
```

III.
```
public static ArrayList<Object> reverse (ArrayList<Object> q)
{
      Object obj;
      if (q.size() != 0)
      {
            obj = q.remove(0);
            q = reverse(q);
            q.add(obj);
      }
      return q;
}
```

Which of the above implementations of method `reverse` work as intended?

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

GO ON TO THE NEXT PAGE.

33. Consider the following code segment.

```
List<Integer> values = new ArrayList<Integer>();
values.add(5);
values.add(3);
values.add(2);
values.add(2);
values.add(6);
values.add(3);
values.add(9);
values.add(2);
values.add(1);
for (int j = 0; j < values.size(); j++)
{
    if (values.get(j).intValue() == 2)
    {
       values.remove(j);
    }
}
```

What will `values` contain as a result of executing this code segment?

(A) `[5, 3, 2, 2, 6, 3, 9, 2, 1]`
(B) `[5, 3, 2, 6, 3, 9, 1]`
(C) `[5, 3, 6, 3, 9, 1]`
(D) `[2, 2, 2, 5, 3, 6, 3, 9, 1]`
(E) The code throws an `ArrayIndexOutOfBoundsException`.

GO ON TO THE NEXT PAGE.

34. Consider the class `Data` partially defined below. The completed `max1D` method returns the maximum value of `b` as a one-dimensional array of integers. The completed `max2D` method is intended to return the maximum value of `c` as a two dimensional array of integers.

```
public class Data
{
   /** Returns the maximum value of one-dimensional array b */
   public int max1D(int[] b)
   { /* implementation not shown */}
   /** Returns the maximum value of two-dimensional array c */
   public int max2D(int[] c)
   {
      int max;
      /* missing code */
      returns max
   }
   /* other methods of Data class not shown*/
}
```

Assume that `max1D` works as intended. Which of the following can replace `/* missing code */` so that `max2D` works as intended?

I.
```
for (int[] row: c)
{
   max = max1D(row);
}
```

II.
```
max = max1D(c[0]);
for (int k = 1; k <= c.length; k++)
{
   max = max1D(c[k]);
}
```

III.
```
max = max1D(c[0]);
for (int[] row: c)
{
   if (max < maxID(row))
   {
      max = max1D(row);
   }
}
```

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

GO ON TO THE NEXT PAGE.

35. Consider the following instance variable, `numList`, and incomplete method, `countZeros`. The method is intended to return an integer array count such that for all `k`, `count[k]` is equal to the number of elements equal to 0 from `numList[0]` through `numList[k]`. For example, if `numList` contains the values {1, 4, 0, 5, 0, 0}, the array `countZeros` contains the values {0, 0, 1, 1, 2, 3}.

```
public int[] countZeros(int[] numList)
{
   int[] count = new int[numList.length];
   for (int k : count)
   {
      count[k] = 0;
   }
   /* missing code */
   return count;
}
```

The following two versions of `/* missing code */` are suggested to make the method work as intended.

Version 1

```
for (int k = 0;  k <= numList.length; k++)
{
   for (int j = 0; j <= k; j++)
   {
        if  (numList[j] == 0)
        {
             count[k] = count[k] + 1;
        }
   }
}
```

Version 2

```
for (int k = 0; k < numList.length; k++)
{
   if  (numList[k] = 0)
   {
        count[k] = count[k - 1] + 1;
   }
   else
   {
        count[k] = count[k - 1];
   }
}
```

Which of the following statements is true?

(A) Both Version 1 and Version 2 will work as intended, but Version 1 is faster than Version 2.
(B) Both Version 1 and Version 2 will work as intended, but Version 2 is faster than Version 1.
(C) Version 1 will work as intended, but Version 2 causes an `ArrayIndexOutOfBoundsException`.
(D) Version 2 will work as intended, but Version 1 causes an `ArrayIndexOutOfBoundsException`.
(E) Version 1 and Version 2 each cause an `ArrayIndexOutOfBoundsException`.

GO ON TO THE NEXT PAGE.

36. A real estate agent wants to develop a program to record information about apartments for rent. For each apartment, she intends to record the number of bedrooms, number of bathrooms, whether pets are allowed, and the monthly rent charged. Which of the following object-oriented program designs would be preferred?

(A) Use a class `Apartment` with four subclasses: `Bedrooms, Bathrooms, PetsAllowed,` and `Rent`.
(B) Use four classes: `Bedrooms, Bathrooms, PetsAllowed,` and `Rent`, each with subclass `Apartment`.
(C) Use a class `Apartment` with four instance variables `int bedrooms, int bathrooms, boolean petsAllowed,` and `double rent.`
(D) Use five unrelated classes: `Apartment, Bedrooms, Bathrooms, PetsAllowed,` and `Rent.`
(E) Use a class `Apartment`, with a subclass `Bedrooms`, with a subclass `Bathrooms`, with a subclass `PetsAllowed`, with a subclass `Rent`.

37. Consider the following declarations:

```
public class Book
{
    boolean hasMorePagesThan(Book b);
    //other methods not shown
}
Public class Dictionary extends Book
{
    //other methods not shown
}
```

Of the following method headings of `hasMorePagesThan`, which can be added to `Dictionary` so that it will satisfy the `Book` superclass?

I. `int hasMorePagesThan(Book b)`
II. `boolean hasMorePagesThan(Book b)`
III. `boolean hasMorePagesThan(Dictionary d)`

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

GO ON TO THE NEXT PAGE.

38. Consider the following method.

```
/**Precondition: set does not contain any negative values
 */
public int mystery(int[] set, int max)
{
    int m = 0;
    int count = 0;
    for (int n = 0; n < set.length && set[n] < max; n++)
    {
        if (set[n] >= m)
        {
            m = set[n]; //Statement A
        }
        count++; //Statement B
    }
    return count;
}
```

Assume that `mystery` is called and is executed without error. Which of the following are possible combinations of the value of `max`, the number of times Statement A is executed, and the number of times Statement B is executed?

	Value of max	Executions of Statement A	Executions of Statement B
I.	8	2	3
II.	3	7	5
III.	7	0	4

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

GO ON TO THE NEXT PAGE.

39. The following method is intended to return an array that inserts an integer `m` at index `n`, pushing the values of all indices after `n` to the index one higher. For example, if the `arr` is

4	2	1	3

and the command `arr = insert(arr, 5, 2)` is called, the method is intended to return

4	2	5	1	3

```
1         public int[] insert(int[] arr, int m, int n)
2         {
3                int[] temp = new int[arr.length+1];
4                for (int k  = 0; k < arr.length; k++)
5                {
6                       if (k < n)
7                       {
8                              temp[k] = arr[k];
9                       }
10                      else
11                      {
12                             temp[k + 1] = arr[k];
13                      }
14               }
15               temp[m] = n;
16               return temp;
17        }
```

The method `insert` does not work as intended. Which of the following changes will cause it to work as intended?

(A) Change Line 6 to `if (k > n)`
(B) Change Line 6 to `if (k <= n)`
(C) Change Line 12 to `temp[k] = arr[k + 1];`
(D) Change Line 15 to `temp[n] = m;`
(E) Change Line 16 to `return arr;`

GO ON TO THE NEXT PAGE.

40. If X, Y, and Z are integer values, the boolean expression

```
(X > Y)  &&  (Y  >  Z)
```

can be replaced by which of the following?

(A) `X  >  Z`

(B) `(X  <  Y)  ||  (Y  <  Z)`

(C) `(X <= Y)  ||  (Y  <= Z)`

(D) `!((X  <  Y)  ||  (Y  <  Z))`

(E) `!((X  <=  Y)  ||  (Y  <=  Z))`

END OF SECTION I

IF YOU FINISH BEFORE TIME IS CALLED,
YOU MAY CHECK YOUR WORK ON THIS SECTION.

DO NOT GO ON TO SECTION II UNTIL YOU ARE TOLD TO DO SO.

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™.

Notes:

- Assume that the classes listed in the Java 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.

FREE-RESPONSE QUESTIONS

1. A monochrome (black-and-white) screen is a rectangular grid of pixels that can be either white or black. A pixel is a location on the screen represented by its row number and column number.

   Consider the following proposal for modeling a screen and its pixels.

   A black pixel on the screen is modeled by an object of type `Pixel`. The `Pixel` class includes the following private data and methods:

   - `row`—this `int` holds the row number of this pixel
   - `col`—this `int` holds the column number of this pixel
   - `Pixel` constructor—this constructor creates a `Pixel` based on the given row and column
   - `getRow`—this method returns the row number of this pixel
   - `getCol`—this method returns the column number of this pixel

```
public class Pixel
{
   private int row;
   private int col;
   public Pixel (int r, int c)
   {row = r; col = c; }
   public int getRow ( )
   {return row; }
   public int getCol ( )
   { return col; }
}
```

GO ON TO THE NEXT PAGE.

A screen is modeled by an object of type `Screen`. Internally, the screen is represented by an array of linked lists of pixels. The index into the array represents the given row on the screen; the linked list at that element represents the *black* pixels at the various columns in order from smallest to largest column. *White pixels are not stored in the linked list.* A pixel not in the list is assumed to be white.

The `Screen` class includes the following private data and methods:

- `data`—The array of linked lists
- `pixelAt`—This method returns the pixel at the given location if it exists (i.e., is black) in this `Screen`. Otherwise, this method returns `null`.
- `pixelOn`—This method creates and stores a black pixel at the appropriate place in the array of linked lists based on the given row and column number.

```
public class Screen
{
    private ArrayList<int>[] data;
    private int numCols;

    // postcondition: data is created with
    //                height elements;
    //                numCols is set to
    //                width
    public Screen (int width, int height)
    {  /* to be implemented in part (a) */

    // precondition: 0 <= row <=
    //               data.length - 1;
    //               0 <= col <= numCols - 1
    // postcondition: returns the pixel at
    //                the given row and col
    //                if it exists (black)
    //                or null if the pixel
    //                doesn't exist (white)
    public Pixel pixelAt (int row, int col)
    { /* to be implemented in part (b) */}

    // precondition: 0 <= row <=
    //               data.length - 1;
    //               0 <= col <= numCols - 1;
    //               the pixel at row, col
    //               does not exist
    //               in this Screen
    // postcondition: adds the pixel at
    //                the given row and col
    //                so that pixels in a
    //                given row of data are
    //                in increasing column
    //                order
    public void pixelOn (int row, int col)
    {  /* to be implemented in part (c) */ }

    //  . . . constructors, other methods,
    //    and other private data not shown
```

GO ON TO THE NEXT PAGE.

(a) Write the constructor for the `Screen` class. The constructor should initialize the private data of the `Screen` class as appropriate.

Complete the constructor for the `Screen` class below.

```
// postcondition: data is created with
//                height elements; numCols
//                is set to width
public Screen(int width, int height)
```

GO ON TO THE NEXT PAGE.

(b) Write the `Screen` method `pixelAt`. Method `pixelAt` should return the pixel at the given row and column of the screen if that pixel exists (i.e., is black). Otherwise `pixelAt` should return `null`.

Complete method `pixelAt` below.

```
// precondition: 0 <= row <=
//               data.length - 1;
//               0 <= col <= numCols - 1
// postcondition: returns the pixel at
//                the given row and col
//                if it exists (black)
//                or null if the pixel
//                doesn't exist (white)
public Pixel pixelAt(int row, int col)
```

GO ON TO THE NEXT PAGE.

(c) Write the `Screen` method `pixelOn`. Method `pixelOn` should modify this `Screen` so that a pixel is stored at the given row and column.

Complete method `pixelOn` below.

```
// precondition: 0 <= row <=
//               data.length - 1;
//               0 <= col <= <= numCols - 1;
//               the pixel at row, col
//               does not exist
//               in this Screen
// postcondition: adds the pixel at
//               the given row and col
//               so that pixels in a
//               given row of a data are
//               in increasing column
//               order
public void pixelOn(int row, int col)
```

GO ON TO THE NEXT PAGE.

2. A toy collector is creating an inventory of her marble collection. A marble set specifies the color and number of a particular group of marbles from her collection. The declaration of the `MarbleSet` class is shown below.

```
public class MarbleSet
{
    /** Constructs a new MarbleSet object */
    public MarbleSet(String color, int numMarbles)
    {  /* implementation not shown */  }

    /** @return the color of the set of marbles
     */
    public String getColor()
    {  /* implementation not shown */  }

    /** @return the number of marbles in the set
      */
     public int getNumber()
     { /* implementation not shown */  }

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

The `MarbleCollection` class documents all sets of marbles in the collection. The declaration of the `MarbleCollection` class is shown below.

```
public class MarbleCollection
{
    /** This is a list of all marble sets */
    private List<MarbleSet> sets;

    /** Constructs a new MarbleSet object */
    public MarbleCollection()
    {  sets = new ArrayList<MarbleSet>();      }

    /** Adds theSet to the marble collection
     * @param theSet the marble set to add to the marble collection
     */
    public void addSet(MarbleSet theSet)
    {  sets.add(theSet);    }

    /** @return the total number of marbles
      */
     public int getTotalMarbles()
     { /* to be implemented in part (a) */   }
    /** Removes all the marble sets from the marble collection that have the same color as
     * marbleColor and returns the total number of marbles removed
     * @param marbleColor the color of the marble sets to be removed
     * @return the total number of marbles of marbleColor in the marble sets removed
     */
    public int removeColor(String marbleCol)
     { /* to be implemented in part (b) */   }
```

GO ON TO THE NEXT PAGE.

(a) The `getTotalMarbles` method computes and returns the sum of the number of marbles. If there are no marble sets, the method returns `0`.

```
Complete method  getTotalMarbles below
/** @return the sum of the number of marbles in all marble sets
 */
public int getTotalMarbles()
```

GO ON TO THE NEXT PAGE.

(b) The method `removeColor` updates the marble collection by removing all the marble sets for which the color of the marbles matches the parameter `marbleCol`. The marble collection may contain zero or more marbles with the same color as the `marbleCol`. The method returns the number of marbles removed.

For example, after the execution of the following code segment

```
MarbleCollection m = new MarbleCollection();
m.addSet(new MarbleSet("red", 2);
m.addSet(new MarbleSet("blue", 3);
m.addSet(new MarbleSet("green", 3);
m.addSet(new MarbleSet("blue", 4);
m.addSet(new MarbleSet("red", 1);
```

the contents of the marble collection can be expressed with the following table.

"red" 2	"blue" 3	"green" 3	"blue" 4	"red" 1

The method call `m.removecolor("red")` returns `3` because there were two red marble sets containing a total of 3 marbles. The new marble collection is shown below.

"blue" 3	"green" 3	"blue" 4

The method call `m.removecolor("purple")` returns 0 and makes no modifications to the marble collection.

Complete the method `removeColor` below.

```
/** Removes all the marble sets from the marble collection that have the same
*color as marbleColor and returns the total number of marbles removed
* @param marbleColor the color of the marble sets to be removed
* @return the total number of marbles of marbleColor in the marble sets removed
*/
     public int removeColor(String marbleCol)
```

GO ON TO THE NEXT PAGE.

3. A binary, or base two, integer is a number consisting of digits that are either 0 or 1. Digits in a binary integer are numbered from right to left starting with 0.

   The decimal value of the binary integer is the sum of each digit multiplied by 2^d where d is the number of the digit.

   For example, the decimal value of the binary integer 1011010 is

   $$(0 * 2^0) + (1 * 2^1) + (0 * 2^2) + (1 * 2^3) + (1 * 2^4) + (0 * 2^5) + (1 + 2^6)$$

   $$= 0 + 2 + 0 + 8 + 16 + 0 + 64$$

   $$= 90$$

   A decimal integer can be converted into its corresponding binary integer according to the following algorithm:

   - Calculate the remainder when the decimal integer is divided by 2. This is the rightmost digit of the corresponding binary integer.
   - Divide the decimal integer by 2 using integer division. If the result is 0, stop. Otherwise, repeat the algorithm using the new value of the decimal integer.

   The digits produced will be in right-to-left order in the binary integer.

   For instance, the decimal integer 90 can be converted into its corresponding binary integer as follows:

90 % 2 = 0		(the rightmost digit)
90 / 2 = 45	45 % 2 = 1	(the second digit from the right)
45 / 2 = 22	22 % 2 = 0	(the third digit from the right)
22 / 2 = 11	11 % 2 = 1	(the fourth digit from the right)
11 / 2 = 55	55 % 2 = 1	(the fifth digit from the right)
5 / 2 = 22	22 % 2 = 0	(the sixth digit from the right)
2 / 2 = 11	11 % 2 = 1	(the leftmost digit)
1 / 2 = 0		

   Consider the design of a class that represents an arbitrary-length non-negative binary integer.

   The operations on this class include

   - constructing an empty binary integer with value zero
   - constructing a binary integer from an arbitrary non-negative decimal integer
   - returning a binary integer that represents the result of adding another binary integer to this binary integer
   - returning the result of converting this binary integer to a `String`
   - returning a positive integer if this binary integer is less than another binary integer, zero if it is equal, and a negative integer if it is less

GO ON TO THE NEXT PAGE.

In addition, the binary integer class should fully implement the `Comparable` interface.

(a) Write the definition of a binary integer class called `BinaryInt`, showing the appropriate data definitions, constructors, and method signatures. You should *not* write the implementations of the constructor or any of the methods you define for the `BinaryInt` class.

(b) Using the signature you wrote in part **(a)**, write the implementation for the operation that constructs a `BinaryInt` from an arbitrary decimal integer. In writing this method, you may call any of the methods in the `BinaryInt` class (as you defined it in part **(a)**). Assume that these methods work as specified.

(c) Using the `BinaryInt` class (as you defined it in part **(a)**), complete the following method, `Test`, that adds the following pairs of decimal integers in binary and outputs the larger of the two binary sums. `Test` is *not* a method of the binary integer class.

Pair 1: 2,314,279,623 and 3,236,550,123. Pair 2: 3,412,579,010 and 2,128,250,735.

In writing this method, you may call any of the methods in `BinaryInt` (as you defined it in part **(a)**). Assume that these methods work as specified.

Complete method `Test` below.

```
public static void Test ()
```

GO ON TO THE NEXT PAGE.

4. A parabola is a graph defined by the equation $y = ax^2 + bx + c$, where a, b, and c are all integers and a is non-zero. The x-value of the axis of symmetry of a parabola is defined by the double $-b/2a$. A point is a pair of integer values, x and y. A point is defined to be on a parabola if it satisfies the equation of the parabola. Consider the examples in the table below:

Equation	Axis of symmetry ($-b/2a$)	Is point (4, 3) on parabola?
$y = 2x^2 - 6x - 5$	$-(-6)/2(2) = 1.5$	Yes, $3 = 2(4)^2 - 6(4) - 5$
$y = 4x^2 + 2x - 3$	$-2/2(4) = -0.25$	No, $3 \neq 4(4)^2 + 2(4) - 3$

The following code segment is from a method outside the class `Parabola` and demonstrates how the `Parabola` class can be used to represent the two equations above:

```
Parabola par1 = new Parabola(2, -6, -5);
double axis1 = par1.getAxis(); //assigns 1.5 to axis1
boolean onGraph1 = par1.isOnGraph(4, 3); //assigns true to onGraph1

Parabola par2 = new Parabola(4, 2, -3);
double axis2 = par2.getAxis(); //assigns -0.25 to axis2
boolean onGraph2 = par2.isOnGraph(4, 3); //assigns false to onGraph2
```

Write the `Parabola` class. The constructor class of `Parabola` must take three integer parameters that represent a, b, and c, successively. You may assume as a precondition that a be a non-zero integer. You must include a `getAxis` method that returns the x-coordinate of the axis of symmetry as a double and an `isOnGraph` method that takes a point represented by two integer parameters, x and y, and returns `true` if the point is on the `Parabola` and returns `false` if it is not. Your class methods must be able to return the values indicated in the examples above. You can ignore any overflow issues.

STOP

END OF EXAM

The Princeton Review®

Completely darken bubbles with a No. 2 pencil. If you make a mistake, be sure to erase mark completely. Erase all stray marks.

1.

YOUR NAME: _____
(Print) Last First M.I.

SIGNATURE: _____ DATE: ___/___/___

HOME ADDRESS: _____
(Print) Number and Street

_____
City State Zip Code

PHONE NO.: _____

IMPORTANT: Please fill in these boxes exactly as shown on the back cover of your test book.

2. TEST FORM

3. TEST CODE

4. REGISTRATION NUMBER

3. TEST CODE					4. REGISTRATION NUMBER					
0	A	J	0	0	0	0	0	0	0	0
1	B	K	1	1	1	1	1	1	1	1
2	C	L	2	2	2	2	2	2	2	2
3	D	M	3	3	3	3	3	3	3	3
4	E	N	4	4	4	4	4	4	4	4
5	F	O	5	5	5	5	5	5	5	5
6	G	P	6	6	6	6	6	6	6	6
7	H	Q	7	7	7	7	7	7	7	7
8	I	R	8	8	8	8	8	8	8	8
9			9	9	9	9	9	9	9	9

6. DATE OF BIRTH

Month	Day		Year	
JAN				
FEB	0	0	0	0
MAR	1	1	1	1
APR	2	2	2	2
MAY	3	3	3	3
JUN		4	4	4
JUL		5	5	5
AUG		6	6	6
SEP		7	7	7
OCT		8	8	8
NOV		9	9	9
DEC				

7. GENDER

- MALE
- FEMALE

The Princeton Review®

5. YOUR NAME

First 4 letters of last name					FIRST INIT	MID INIT
A	A	A	A		A	A
B	B	B	B		B	B
C	C	C	C		C	C
D	D	D	D		D	D
E	E	E	E		E	E
F	F	F	F		F	F
G	G	G	G		G	G
H	H	H	H		H	H
I	I	I	I		I	I
J	J	J	J		J	J
K	K	K	K		K	K
L	L	L	L		L	L
M	M	M	M		M	M
N	N	N	N		N	N
O	O	O	O		O	O
P	P	P	P		P	P
Q	Q	Q	Q		Q	Q
R	R	R	R		R	R
S	S	S	S		S	S
T	T	T	T		T	T
U	U	U	U		U	U
V	V	V	V		V	V
W	W	W	W		W	W
X	X	X	X		X	X
Y	Y	Y	Y		Y	Y
Z	Z	Z	Z		Z	Z

1.	A	B	C	D	E	11.	A	B	C	D	E	21.	A	B	C	D	E	31.	A	B	C	D	E
2.	A	B	C	D	E	12.	A	B	C	D	E	22.	A	B	C	D	E	32.	A	B	C	D	E
3.	A	B	C	D	E	13.	A	B	C	D	E	23.	A	B	C	D	E	33.	A	B	C	D	E
4.	A	B	C	D	E	14.	A	B	C	D	E	24.	A	B	C	D	E	34.	A	B	C	D	E
5.	A	B	C	D	E	15.	A	B	C	D	E	25.	A	B	C	D	E	35.	A	B	C	D	E
6.	A	B	C	D	E	16.	A	B	C	D	E	26.	A	B	C	D	E	36.	A	B	C	D	E
7.	A	B	C	D	E	17.	A	B	C	D	E	27.	A	B	C	D	E	37.	A	B	C	D	E
8.	A	B	C	D	E	18.	A	B	C	D	E	28.	A	B	C	D	E	38.	A	B	C	D	E
9.	A	B	C	D	E	19.	A	B	C	D	E	29.	A	B	C	D	E	39.	A	B	C	D	E
10.	A	B	C	D	E	20.	A	B	C	D	E	30.	A	B	C	D	E	40.	A	B	C	D	E