

1. Consider the following code segment.

```
int value = 15;
while (value < 28)
{
    System.out.println(value);
    value++;
}
```

What are the first and last numbers output by the code segment?

- |     | <u>First</u> | <u>Last</u> |
|-----|--------------|-------------|
| (A) | 15           | 27          |
| (B) | 15           | 28          |
| (C) | 16           | 27          |
| (D) | 16           | 28          |
| (E) | 16           | 29          |

2. A teacher put three bonus questions on a test and awarded 5 extra points to anyone who answered all three bonus questions correctly and no extra points otherwise. Assume that the boolean variables `bonusOne`, `bonusTwo`, and `bonusThree` indicate whether a student has answered the particular question correctly. Each variable was assigned `true` if the answer was correct and `false` if the answer was incorrect.

Which of the following code segments will properly update the variable `grade` based on a student's performance on the bonus questions?

- I. 

```
if (bonusOne && bonusTwo && bonusThree)
    grade += 5;
```
- II. 

```
if (bonusOne || bonusTwo || bonusThree)
    grade += 5;
```
- III. 

```
if (bonusOne)
    grade += 5;
if (bonusTwo)
    grade += 5;
if (bonusThree)
    grade += 5;
```

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

3. Assume that an array of integer values has been declared as follows and has been initialized.

```
int[] arr = new int[10];
```

Which of the following code segments correctly interchanges the value of `arr[0]` and `arr[5]` ?

- (A) `arr[0] = 5;`  
`arr[5] = 0;`
- (B) `arr[0] = arr[5];`  
`arr[5] = arr[0];`
- (C) `int k = arr[5];`  
`arr[0] = arr[5];`  
`arr[5] = k;`
- (D) `int k = arr[0];`  
`arr[0] = arr[5];`  
`arr[5] = k;`
- (E) `int k = arr[5];`  
`arr[5] = arr[0];`  
`arr[0] = arr[5];`

4. Consider the following code segment.

```
ArrayList<String> items = new ArrayList<String>();  
items.add("A");  
items.add("B");  
items.add("C");  
items.add(0, "D");  
items.remove(3);  
items.add(0, "E");  
System.out.println(items);
```

What is printed as a result of executing the code segment?

- (A) [A, B, C, E]
  - (B) [A, B, D, E]
  - (C) [E, D, A, B]
  - (D) [E, D, A, C]
  - (E) [E, D, C, B]
- 
5. When designing a class hierarchy, which of the following should be true of a superclass?
- (A) A superclass should contain the data and functionality that are common to all subclasses that inherit from the superclass.
  - (B) A superclass should be the largest, most complex class from which all other subclasses are derived.
  - (C) A superclass should contain the data and functionality that are only required for the most complex class.
  - (D) A superclass should have public data in order to provide access for the entire class hierarchy.
  - (E) A superclass should contain the most specific details of the class hierarchy.

Questions 6-7 refer to the following code segment.

```
int k = a random number such that  $1 \leq k \leq n$  ;  
  
for (int p = 2; p <= k; p++)  
    for (int r = 1; r < k; r++)  
        System.out.println("Hello");
```

6. What is the minimum number of times that Hello will be printed?

- (A) 0
- (B) 1
- (C) 2
- (D)  $n - 1$
- (E)  $n - 2$

7. What is the maximum number of times that Hello will be printed?

- (A) 2
- (B)  $n - 1$
- (C)  $n - 2$
- (D)  $(n - 1)^2$
- (E)  $n^2$

**Section I**

8. Consider the following instance variable and incomplete method. The method `calcTotal` is intended to return the sum of all values in `vals`.

```
private int[] vals;

public int calcTotal()
{
    int total = 0;

    /* missing code */

    return total;
}
```

Which of the code segments shown below can be used to replace `/* missing code */` so that `calcTotal` will work as intended?

- I. 

```
for (int pos = 0; pos < vals.length; pos++)
{
    total += vals[pos];
}
```
- II. 

```
for (int pos = vals.length; pos > 0; pos--)
{
    total += vals[pos];
}
```
- III. 

```
int pos = 0;
while (pos < vals.length)
{
    total += vals[pos];
    pos++;
}
```

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

9. Consider the following code segment.

```
String str = "abcdef";  
for (int rep = 0; rep < str.length() - 1; rep++)  
{  
    System.out.print(str.substring(rep, rep + 2));  
}
```

What is printed as a result of executing this code segment?

- (A) abcdef
- (B) aabbccddeeff
- (C) abbccddeef
- (D) abcbcdcdedef
- (E) Nothing is printed because an `IndexOutOfBoundsException` is thrown.

10. Consider the following method.

```
public void numberCheck(int maxNum)
{
    int typeA = 0;
    int typeB = 0;
    int typeC = 0;

    for (int k = 1; k <= maxNum; k++)
    {
        if (k % 2 == 0 && k % 5 == 0)
            typeA++;
        if (k % 2 == 0)
            typeB++;
        if (k % 5 == 0)
            typeC++;
    }

    System.out.println(typeA + " " + typeB + " " + typeC);
}
```

What is printed as a result of the call `numberCheck(50)` ?

- (A) 5 20 5
- (B) 5 20 10
- (C) 5 25 5
- (D) 5 25 10
- (E) 30 25 10



11. Consider the following method that is intended to modify its parameter `nameList` by replacing all occurrences of `name` with `newValue`.

```
public void replace(ArrayList<String> nameList,  
                   String name, String newValue)  
{  
    for (int j = 0; j < nameList.size(); j++)  
    {  
        if ( /* expression */ )  
        {  
            nameList.set(j, newValue);  
        }  
    }  
}
```

Which of the following can be used to replace `/* expression */` so that `replace` will work as intended?

- (A) `nameList.get(j).equals(name)`
- (B) `nameList.get(j) == name`
- (C) `nameList.remove(j)`
- (D) `nameList[j] == name`
- (E) `nameList[j].equals(name)`

12. Consider the following incomplete method.

```
public int someProcess(int n)
{
    /* body of someProcess */
}
```

The following table shows several examples of input values and the results that should be produced by calling someProcess.

Input Value n	Value Returned by someProcess(n)
3	30
6	60
7	7
8	80
9	90
11	11
12	120
14	14
16	160

Which of the following code segments could be used to replace */\* body of someProcess \*/* so that the method will produce the results shown in the table?

- I. 

```
if ((n % 3 == 0) && (n % 4 == 0))
    return n * 10;
else
    return n;
```
- II. 

```
if ((n % 3 == 0) || (n % 4 == 0))
    return n * 10;

return n;
```

```
III.  if (n % 3 == 0)
      if (n % 4 == 0)
          return n * 10;

      return n;
```

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

13. Consider the following method.

```
// precondition: x >= 0
public void mystery(int x)
{
    if ((x / 10) != 0)
    {
        mystery(x / 10);
    }

    System.out.print(x % 10);
}
```

Which of the following is printed as a result of the call `mystery(123456)` ?

- (A) 16
- (B) 56
- (C) 123456
- (D) 654321
- (E) Many digits are printed due to infinite recursion.

14. Consider the following instance variables and incomplete method that are part of a class that represents an item. The variables `years` and `months` are used to represent the age of the item, and the value for `months` is always between 0 and 11, inclusive. Method `updateAge` is used to update these variables based on the parameter `extraMonths` that represents the number of months to be added to the age.

```
private int years;
private int months; // 0 <= months <= 11

// precondition: extraMonths >= 0
public void updateAge(int extraMonths)
{
    /* body of updateAge */
}
```

Which of the following code segments could be used to replace `/* body of updateAge */` so that the method will work as intended?

- I. `int yrs = extraMonths % 12;`  
`int mos = extraMonths / 12;`  
`years = years + yrs;`  
`months = months + mos;`
- II. `int totalMonths = years * 12 + months + extraMonths;`  
`years = totalMonths / 12;`  
`months = totalMonths % 12;`
- III. `int totalMonths = months + extraMonths;`  
`years = years + totalMonths / 12;`  
`months = totalMonths % 12;`

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

15. Consider the following method.

```
public String inRangeMessage(int value)
{
    if (value < 0 || value > 100)
        return "Not in range";
    else
        return "In range";
}
```

Consider the following code segments that could be used to replace the body of `inRangeMessage`.

- I. 

```
if (value < 0)
{
    if (value > 100)
        return "Not in range";
    else
        return "In range";
}
else
    return "In range";
```
- II. 

```
if (value < 0)
    return "Not in range";
else if (value > 100)
    return "Not in range";
else
    return "In range";
```
- III. 

```
if (value >= 0)
    return "In range";
else if (value <= 100)
    return "In range";
else
    return "Not in range";
```

Which of the replacements will have the same behavior as the original version of `inRangeMessage` ?

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

16. Consider the following class declaration.

```
public class SomeClass
{
    private int num;

    public SomeClass(int n)
    {
        num = n;
    }

    public void increment(int more)
    {
        num = num + more;
    }

    public int getNum()
    {
        return num;
    }
}
```

The following code segment appears in another class.

```
SomeClass one = new SomeClass(100);
SomeClass two = new SomeClass(100);
SomeClass three = one;

one.increment(200);

System.out.println(one.getNum() + " " + two.getNum() + " " +
                    three.getNum());
```

What is printed as a result of executing the code segment?

- (A) 100 100 100
- (B) 300 100 100
- (C) 300 100 300
- (D) 300 300 100
- (E) 300 300 300

17. The following incomplete method is intended to sort its array parameter `arr` in increasing order.

```
// postcondition: arr is sorted in increasing order
public static void sortArray(int[] arr)
{
    int j, k;

    for (j = arr.length - 1; j > 0; j--)
    {
        int pos = j;

        for ( /* missing code */ )
        {
            if (arr[k] > arr[pos])
            {
                pos = k;
            }
        }
        swap(arr, j, pos);
    }
}
```

Assume that `swap(arr, j, pos)` exchanges the values of `arr[j]` and `arr[pos]`. Which of the following could be used to replace `/* missing code */` so that executing the code segment sorts the values in array `arr`?

- (A) `k = j - 1; k > 0; k--`
- (B) `k = j - 1; k >= 0; k--`
- (C) `k = 1; k < arr.length; k++`
- (D) `k = 1; k > arr.length; k++`
- (E) `k = 0; k <= arr.length; k++`



---

18. Assume that  $x$  and  $y$  are boolean variables and have been properly initialized.

$$(x \ \&\& \ y) \ || \ !(x \ \&\& \ y)$$

The result of evaluating the expression above is best described as

- (A) always true
- (B) always false
- (C) true only when  $x$  is true and  $y$  is true
- (D) true only when  $x$  and  $y$  have the same value
- (E) true only when  $x$  and  $y$  have different values

19. Assume that the following variable declarations have been made.

```
double d = Math.random();  
double r;
```

Which of the following assigns a value to  $r$  from the uniform distribution over the range  $0.5 \leq r < 5.5$  ?

- (A)  $r = d + 0.5;$
- (B)  $r = d + 0.5 * 5.0;$
- (C)  $r = d * 5.0;$
- (D)  $r = d * 5.0 + 0.5;$
- (E)  $r = d * 5.5;$

20. Consider the following instance variables and method that appear in a class representing student information.

```
private int assignmentsCompleted;
private double testAverage;

public boolean isPassing()
{ /* implementation not shown */ }
```

A student can pass a programming course if at least one of the following conditions is met.

- The student has a test average that is greater than or equal to 90.
- The student has a test average that is greater than or equal to 75 and has at least 4 completed assignments.

Consider the following proposed implementations of the `isPassing` method.

I. 

```
if (testAverage >= 90)
    return true;
if (testAverage >= 75 && assignmentsCompleted >= 4)
    return true;
return false;
```

II. 

```
boolean pass = false;
if (testAverage >= 90)
    pass = true;
if (testAverage >= 75 && assignmentsCompleted >= 4)
    pass = true;
return pass;
```

III. 

```
return (testAverage >= 90) ||
        (testAverage >= 75 && assignmentsCompleted >= 4);
```

Which of the implementations will correctly implement method `isPassing`?

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

Questions 21-25 refer to the code from the GridWorld case study. A copy of the code is provided in the Appendix.

21. Consider the following code segment.

```
Location loc1 = new Location(3, 3);
Location loc2 = new Location(3, 2);

if (loc1.equals(loc2.getAdjacentLocation(Location.EAST)))
    System.out.print("aaa");

if (loc1.getRow() == loc2.getRow())
    System.out.print("XXX");

if (loc1.getDirectionToward(loc2) == Location.EAST)
    System.out.print("555");
```

What will be printed as a result of executing the code segment?

- (A) aaaXXX555
- (B) aaaXXX
- (C) XXX555
- (D) 555
- (E) aaa

22. A `RightTurningBug` behaves like a `Bug`, except that when it turns, it turns 90 degrees to the right. The declaration for the `RightTurningBug` class is as follows.

```
public class RightTurningBug extends Bug
{
    public void turn()
    {
        /* missing implementation */
    }
}
```

Consider the following suggested replacements for `/* missing implementation */`.

I. `int desiredDirection = (getDirection() + Location.RIGHT) % Location.FULL_CIRCLE;`

```
while (getDirection() != desiredDirection)
{
    super.turn();
}
```

II. `super.turn();`  
`super.turn();`

III. `setDirection(getDirection() + Location.RIGHT);`

Which of the replacements will produce the desired behavior?

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

23. Consider the following declarations.

```
Actor a = new Actor();  
Bug b = new Bug();  
Rock r = new Rock();  
Critter c = new Critter();
```

Consider the following lines of code.

```
Line 1:  int dir1 = c.getDirection();  
Line 2:  int dir2 = a.getDirection();  
Line 3:  int dir3 = b.getDirection();  
Line 4:  ArrayList<Location> rLoc = r.getMoveLocations();  
Line 5:  ArrayList<Location> cLoc = c.getMoveLocations();
```

Which of the lines of code above will cause a compile time error?

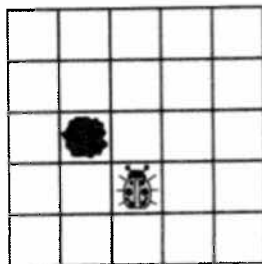
- (A) Line 1 only
- (B) Lines 2 and 3 only
- (C) Line 4 only
- (D) Line 5 only
- (E) Lines 4 and 5 only

24. Consider the following `TestBug` class declaration.

```
public class TestBug extends Bug
{
    public void act()
    {
        if (canMove())
        {
            move();
            if (canMove())
                move();
        }
        else
        {
            setDirection(getDirection() + Location.HALF_CIRCLE);
        }
    }
}
```

The following code segment will produce a grid that has a `Rock` object and a `TestBug` object placed as shown.

```
Grid<Actor> g = new BoundedGrid<Actor>(5, 5);
Rock r = new Rock();
r.putSelfInGrid(g, new Location(2, 1));
Bug t = new TestBug();
t.putSelfInGrid(g, new Location(3, 2));
```



Which of the following best describes what the `TestBug` object `t` does as a result of calling `t.act()` ?

- (A) Moves forward two locations and remains facing current direction
- (B) Moves forward two locations and turns 180 degrees
- (C) Moves forward one location and remains facing current direction
- (D) Moves forward one location and turns 180 degrees
- (E) Stays in the same location and turns 180 degrees

25. A `DancingCritter` is a `Critter` that moves in the following manner. The `DancingCritter` makes a left turn if at least one of its neighbors is another `DancingCritter`. It then moves like a `Critter`. If none of its neighbors are `DancingCritter` objects, it moves like a `Critter` without making a left turn. In all other respects, a `DancingCritter` acts like a `Critter` by eating neighbors that are not rocks or critters. Consider the following implementations.

```
I. public class DancingCritter extends Critter
{
    public ArrayList<Actor> getActors()
    {
        ArrayList<Actor> actors = new ArrayList<Actor>();
        for (Actor a : getGrid().getNeighbors(getLocation()))
        {
            if (a instanceof DancingCritter)
                actors.add(a);
        }
        return actors;
    }

    public void processActors(ArrayList<Actor> actors)
    {
        if (actors.size() > 0)
        {
            setDirection(getDirection() + Location.LEFT);
        }
        super.processActors(actors);
    }
}
```

```
II. public class DancingCritter extends Critter
{
    public void processActors(ArrayList<Actor> actors)
    {
        boolean turning = false;
        for (Actor a : actors)
        {
            if (a instanceof DancingCritter)
                turning = true;
        }
        if (turning)
        {
            setDirection(getDirection() + Location.LEFT);
        }
    }
}
```



```
III. public class DancingCriticter extends Critter
{
    public void makeMove(Location loc)
    {
        boolean turning = false;
        for (Actor a : getGrid().getNeighbors(getLocation()))
        {
            if (a instanceof DancingCriticter)
                turning = true;
        }
        if (turning)
        {
            setDirection(getDirection() + Location.LEFT);
        }
        super.makeMove(loc);
    }
}
```

Which of the proposed implementations will correctly implement the DancingCriticter class?

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

26. Consider the following code segment.

```
int k = 0;
while (k < 10)
{
    System.out.print((k % 3) + " ");
    if ((k % 3) == 0)
        k = k + 2;
    else
        k++;
}
```

What is printed as a result of executing the code segment?

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

27. Consider the following method. Method `allEven` is intended to return `true` if all elements in array `arr` are even numbers; otherwise, it should return `false`.

```
public boolean allEven(int[] arr)
{
    boolean isEven = /* expression */ ;

    for (int k = 0; k < arr.length; k++)
    {
        /* loop body */
    }

    return isEven;
}
```

Which of the following replacements for `/* expression */` and `/* loop body */` should be used so that method `allEven` will work as intended?

- |     | <u>/* expression */</u> | <u>/* loop body */</u>  |
|-----|-------------------------|---|
| (A) | <code>false</code>      | <code>if ((arr[k] % 2) == 0)<br/>isEven = true;</code>                              |
| (B) | <code>false</code>      | <code>if ((arr[k] % 2) != 0)<br/>isEven = false;<br/>else<br/>isEven = true;</code> |
| (C) | <code>true</code>       | <code>if ((arr[k] % 2) != 0)<br/>isEven = false;</code>                             |
| (D) | <code>true</code>       | <code>if ((arr[k] % 2) != 0)<br/>isEven = false;<br/>else<br/>isEven = true;</code> |
| (E) | <code>true</code>       | <code>if ((arr[k] % 2) == 0)<br/>isEven = false;<br/>else<br/>isEven = true;</code> |

28. Consider the following code segment.

```
int x = /* some integer value */ ;
int y = /* some integer value */ ;

boolean result = (x < y);

result = ( (x >= y) && !result );
```

Which of the following best describes the conditions under which the value of `result` will be true after the code segment is executed?

- (A) Only when  $x < y$
- (B) Only when  $x \geq y$
- (C) Only when  $x$  and  $y$  are equal
- (D) The value will always be true.
- (E) The value will never be true.

29. Consider the following code segment.

```
for (int outer = 0; outer < n; outer++)
{
    for (int inner = 0; inner <= outer; inner++)
    {
        System.out.print(outer + " ");
    }
}
```

If `n` has been declared as an integer with the value 4, what is printed as a result of executing the code segment?

- (A) 0 1 2 3
- (B) 0 0 1 0 1 2
- (C) 0 1 2 2 3 3 3
- (D) 0 1 1 2 2 2 3 3 3 3
- (E) 0 0 1 0 1 2 0 1 2 3

30. Consider the following class declarations.

```
public class Base
{
    private int myVal;

    public Base()
    { myVal = 0; }

    public Base(int x)
    { myVal = x; }
}

public class Sub extends Base
{
    public Sub()
    { super(0); }
}
```

Which of the following statements will NOT compile?

- (A) Base b1 = new Base();
- (B) Base b2 = new Base(5);
- (C) Base s1 = new Sub();
- (D) Sub s2 = new Sub();
- (E) Sub s3 = new Sub(5);

31. Assume that `a` and `b` are variables of type `int`. The expression

```
!(a < b) && !(a > b)
```

is equivalent to which of the following?

- (A) true
- (B) false
- (C) `a == b`
- (D) `a != b`
- (E) `!(a < b) && (a > b)`

32. Consider the following code segment.

```
int a = 24;
int b = 30;
while (b != 0)
{
    int r = a % b;
    a = b;
    b = r;
}
```

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

What is printed as a result of executing the code segment?

- (A) 0
- (B) 6
- (C) 12
- (D) 24
- (E) 30

33. Consider the following method.

```
public int sol(int lim)
{
    int s = 0;

    for (int outer = 1; outer <= lim; outer++)
    {
        for (int inner = outer; inner <= lim; inner++)
        {
            s++;
        }
    }

    return s;
}
```

What value is returned as a result of the call `sol(10)` ?

- (A) 20
- (B) 45
- (C) 55
- (D) 100
- (E) 385

34. Consider the following incomplete method. Method `findNext` is intended to return the index of the first occurrence of the value `val` beyond the position `start` in array `arr`.

```
// returns index of first occurrence of val in arr
// after position start;
// returns arr.length if val is not found
public int findNext(int[] arr, int val, int start)
{
    int pos = start + 1;

    while ( /* condition */ )
        pos++;

    return pos;
}
```

For example, consider the following code segment.

```
int[] arr = {11, 22, 100, 33, 100, 11, 44, 100};
System.out.println(findNext(arr, 100, 2));
```

The execution of the code segment should result in the value 4 being printed.

Which of the following expressions could be used to replace `/* condition */` so that `findNext` will work as intended?

- (A) `(pos < arr.length) && (arr[pos] != val)`
- (B) `(arr[pos] != val) && (pos < arr.length)`
- (C) `(pos < arr.length) || (arr[pos] != val)`
- (D) `(arr[pos] == val) && (pos < arr.length)`
- (E) `(pos < arr.length) || (arr[pos] == val)`



35. Consider the following code segments.

- I. 

```
int k = 1;
while (k < 20)
{
    if (k % 3 == 1)
        System.out.print( k + " ");

    k = k + 3;
}
```
- II. 

```
for (int k = 1; k < 20; k++)
{
    if (k % 3 == 1)
        System.out.print( k + " ");
}
```
- III. 

```
for (int k = 1; k < 20; k = k + 3)
{
    System.out.print( k + " ");
}
```

Which of the code segments above will produce the following output?

1 4 7 10 13 16 19

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

36. Consider the following two methods that appear within a single class.

```
public void changeIt(int[] list, int num)
{
    list = new int[5];
    num = 0;

    for (int x = 0; x < list.length; x++)
        list[x] = 0;
}
```

```
public void start()
{
    int[] nums = {1, 2, 3, 4, 5};
    int value = 6;

    changeIt(nums, value);

    for (int k = 0; k < nums.length; k++)
        System.out.print(nums[k] + " ");

    System.out.print(value);
}
```

What is printed as a result of the call `start()` ?

- (A) 0 0 0 0 0 0
- (B) 0 0 0 0 0 6
- (C) 1 2 3 4 5 6
- (D) 1 2 3 4 5 0
- (E) `changeIt` will throw an exception.

37. Consider the following declaration of the class `NumSequence`, which has a constructor that is intended to initialize the instance variable `seq` to an `ArrayList` of `numberOfValues` random floating-point values in the range `[0.0, 1.0)`.

```
public class NumSequence
{
    private ArrayList<Double> seq;

    // precondition: numberOfValues > 0
    // postcondition: seq has been initialized to an ArrayList of
    //                length numberOfValues; each element of seq
    //                contains a random Double in the range [0.0, 1.0)
    public NumSequence(int numberOfValues)
    {
        /* missing code */
    }
}
```

Which of the following code segments could be used to replace `/* missing code */` so that the constructor will work as intended?

- I. `ArrayList<Double> seq = new ArrayList<Double>();`  
`for (int k = 0; k < numberOfValues; k++)`  
 `seq.add(new Double(Math.random()));`
- II. `seq = new ArrayList<Double>();`  
`for (int k = 0; k < numberOfValues; k++)`  
 `seq.add(new Double(Math.random()));`
- III. `ArrayList<Double> temp = new ArrayList<Double>();`  
`for (int k = 0; k < numberOfValues; k++)`  
 `temp.add(new Double(Math.random()));`  
  
`seq = temp;`

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

38. Consider the following code segment.

```
double a = 1.1;
double b = 1.2;

if ((a + b) * (a - b) != (a * a) - (b * b))
{
    System.out.println("Mathematical error!");
}
```

Which of the following best describes why the phrase "Mathematical error!" would be printed? (Remember that mathematically  $(a + b) * (a - b) = a^2 - b^2$ .)

- (A) Precedence rules make the `if` condition true.
- (B) Associativity rules make the `if` condition true.
- (C) Roundoff error makes the `if` condition true.
- (D) Overflow makes the `if` condition true.
- (E) A compiler bug or hardware error has occurred.

39. Consider the following recursive method.

```
public static String recur(int val)
{
    String dig = "" + (val % 3);

    if (val / 3 > 0)
        return dig + recur(val / 3);

    return dig;
}
```

What is printed as a result of executing the following statement?

```
System.out.println(recur(32));
```

- (A) 20
- (B) 102
- (C) 210
- (D) 1020
- (E) 2101

40. Consider the following method.

```
public String goAgain(String str, int index)
{
    if (index >= str.length())
        return str;

    return str + goAgain(str.substring(index), index + 1);
}
```

What is printed as a result of executing the following statement?

```
System.out.println(goAgain("today", 1));
```

- (A) today
- (B) todayto
- (C) todayoday
- (D) todayodayay
- (E) todayodaydayayy

**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.**

---

---

**NO TEST MATERIAL ON THIS PAGE**