Multiple Choice

NAME:

DATE:

1. Consider the following code segment.

```
x = 6;
y = 19;
z = 2;
if (x > y)
   if (z > x)
      z++;
else
   z -= 5;
y += x;
```

After this code is executed, the values of `x`, `y`, and `z` are:

a. `x = 6, y = 25, z = -3`
b. `x = 6, y = 19, z = 2`
c. `x = 6, y = 25, z = 3`
d. `x = 6, y = 25, z = 2`
e. `x = 6, y = 19, z = 3`

2. Assume the following declarations have been made.

```
String fName = "Robert";
String lName = "Roberts";
```

Which of the following code segments returns **true**?

I. `return (fName == lName.substring(0, fName.length()));`
II. `return (fName.equals(lName.substring(0, fName.length())));`
III. `return (fName.equals(lName.substring(0, lName.length())));`

a. I only
b. II only
c. III only
d. I and II only
e. II and III only

3. Consider the following code segment.

```
int x = some positive integer value
double a = x;
double c = Math.pow(Math.sqrt(a), 2);
return (a == c);
```

Which of the following statements about the code segment is **true**?

a. A compile-time error occurs because `x` is not a double.
b. A compile-time error occurs because a method call cannot be a parameter to a method.
c. A run-time error occurs.
d. `true` is returned.
e. `true` or `false` may be returned, depending on roundoff errors.

4. Consider the following segment of code.

```
String word = "compute";
int len = word.length();
int num = 3;
String foo = word.substring(len % num, num);
String hoo = word.substring(num + 1);
if (foo.compareTo(hoo) < 0)
{
   hoo += foo;
}
else
{
   foo += hoo;
}
```

After this code is executed, what are the values of `foo` and `hoo`?

a. `foo` = "om", `hoo` = "uteom"
b. `foo` = "ute", `hoo` = "omute"
c. `foo` = "ute", `hoo` = "uteom"
d. `foo` = "m", `hoo` = "utem"
e. `foo` = "ute", `hoo` = "mute"

5. In a simplified game of craps you roll 2 dice. If you get a 7 or an 11 on the *first* roll, you win; otherwise the game continues. Given the following declarations,

```
int diceSum;
int numRolls;
```

where `diceSum` holds the sum of the two dice rolled and `numRolls` holds the number of times the dice were rolled, which of the following code segments tests for a *win* and prints *WIN* to the screen if the condition to win, as defined above, is satisfied? Assume `diceSum` and `numRolls` are assigned their values before this code is executed.

I.
```
if (numRolls == 1 && diceSum == 7 || diceSum == 11)
   System.out.println("WIN");
else
   // Game continues
```

II.
```
if (numRolls == 1 && (diceSum == 7 || diceSum == 11))
   System.out.println("WIN");
else
   // Game continues
```

III.
```
if (numRolls != 1 || (diceSum != 7 && diceSum != 11))
   // Game continues
else
   System.out.println("WIN");
```

a. I only
b. II only
c. III only
d. II and III only
e. I and III only

6. Consider the following code segment.

```
int sum = 200;
int n = 0;
if ((n != 0) && (sum / n > 90))
   return sum += sum;
else
   return sum;
```

What is the result when this code is executed?

a. A run-time error occurs when evaluating `sum / n`.
b. A compile-time error occurs when evaluating `sum / n`.
c. `0` is returned.
d. `200` is returned.
e. `400` is returned.

7. Consider the following code segment.

```
String string1 = "Hello";
String string2 = "World";
string2 = string1;
string1 += "There!";
```

After the code is executed, what are the values of `string1` and `string2`?

a. `string1 = "Hello", string2 = "World"`
b. `string1 = "HelloWorld", string2 = "World"`
c. `string1 = "HelloThere!", string2 = "HelloThere!"`
d. `string1 = "HelloThere!", string2 = "World"`
e. `string1 = "HelloThere!", string2 = "Hello"`

Questions 8 and 9 refer to the `Point` class partially defined below.

```
public class Point
{

   /**
      Constructs a point.
      @param x1 the x-value of the point
      @param y1 the y-value of the point
   */
   Point(int x1, int y1)
   {
      x = x1;
      y = y1;
   }

   /**
      Sets new Point coordinates.
      @param num1 the new x-coordinate
      @param num2 the new y-coordinate
   */
   public void setPoint(int num1, int num2)
   {
      x = num1;
      y = num2;
   }

   /**
      Returns x-coordinate of Point.
      @return the x-coordinate of the point
   */
   public int getX()
   {
      return x;
   }

   /**
      Returns y-coordinate of Point.
      @return the y-coordinate of the point
   */
   public int getY()
   {
      return y;
   }

   /**
      Returns true if the x-coordinate of this point has the same
      value as its y-coordinate. Returns false otherwise.
      @return true if x and y coordinates are equal, otherwise
      returns false.
   */
   public boolean hasSameXandY()
   {
      // Code goes here
   }

   private int x;    // x-coordinate of point
   private int y;    // y-coordinate of point
}
```

8. A client program needs to check to see whether two `Points` have the same x- and y-coordinates. Given that the object variables `p1` and `p2` refer to different objects that have the same x- and y- coordinates, which of the following code segments will test that equality?

I.
```
if (p1.x == p2.x && p1.y == p2.y)
    statement;
```

II.
```
if (p1.equals(p2))
    statement;
```

III.
```
if (p1.getX() == p2.getX() && p1.getY() == p2.getY())
    statement;
```

a. I only
b. II only
c. III only
d. I and II only
e. II and III only

9. Which of the following code segments could replace the body of the `Point` method `hasSameXandY` so that the method returns `true` if the point's x-coordinate is the same value as its y-coordinate and returns `false` otherwise.

I.
```
return (x == y);
```

II.
```
if ((getX()).equals(getY()))
    return true;
else
    return false;
```

III.
```
if (getX() == getY())
    return true;
else
    return false;
```

a. I only
b. II only
c. III only
d. I and II only
e. I and III only

10. The Boolean expression `!(!A && !B)` is equivalent to

a. `(!A && !B)`
b. `(!A || !B)`
c. `(A || B)`
d. `!(A || B)`
e. `(!A || B)`

11. Which of the following is **true** about unit testing?

a. A unit test simultaneously checks all classes that are included in a program.
b. A unit test simultaneously checks a set of cooperating classes in a program.
c. A unit test simultaneously checks all of the methods of one class.
d. A unit test checks a single method or set of cooperating methods in a class.
e. A unit test checks a single statement in a method.

Questions 12–14 refer to the following problem.

The method `weeklyPay` is to return the amount of pay an employee will receive for one week's work. The parameters for `weeklyPay` are the hours worked during the week and the hourly pay. The weekly pay is based on the following standards. If the employee worked 40 hours or less during the week, he receives his regular hourly rate times the number of hours he worked. For every hour over 40 hours, the employee receives time and a half (1.5 times his regular hourly pay). The following information and method header is given.

```
final int MAX_HOURS_IN_WEEK = 168;   // 7 * 24

/**
   Computes weekly pay.
   Assumes 0 <= hoursWorked and
   hoursWorked <= MAX_HOURS_IN_WEEK, 0 < hourlyPay
   @param hoursWorked is hours worked
   @param hourlyPay is pay per hour
*/
public static double weeklyPay(double hoursWorked,
      double hourlyPay)
```

12. Which is the best set of test values for the variable `hoursWorked`?

    a. 0, 40, 168
    b. 5, 10, 20, 30, 40, 170
    c. −1, 0, 1, 39, 40, 41, 167, 168, 169
    d. −10, 20, 30, 40, 50, 200
    e. −20, −10, 10, 20, 30, 40, 100, 200

13. Which of the following method calls would test illegal input?

    I. `weeklyPay(0, 0)`
    II. `weeklyPay(168, 169)`
    III. `weeklyPay(0, 200)`

    a. I only
    b. II only
    c. III only
    d. I and II only
    e. I and III only

14. Which of the method calls would be good choices to test boundary conditions or branches of conditionals in the `weeklyPay` method?

    I. `weeklyPay(0, 10)`
    II. `weeklyPay(168, 20)`
    III. `weeklyPay(40, 40)`

    a. I only
    b. II only
    c. III only
    d. I and II only
    e. I, II, and III