Sample Multiple-Choice Question 1 Solution

01. (A)

You always need to be very careful when a loop traversal removes items from the very array it is traversing. The **size** of **list** changes dynamically during program execution.

k	list.get(k)	list.get(k) % 2	list
			0: 23, 1: 39, 2: 42, 3: 28, 4: 99, 5: 57
0	23	1	0: 39, 1: 42, 2: 28, 3: 99, 4: 57
1	42	0	
2	28	0	
3	99	1	0: 39, 1: 42, 2: 28, 3: 57

Sample Multiple-Choice Question 2 Solution

02. (C)

There are $\mathbf{2^{32}}$ possible **int** values. Half of those are used for the negative integers. The other half are used for the positive integers, and **0**. This means we have 1 fewer positive number. If we focus on the absolute values, the largest positive integer (**Integer.MAX_VALUE**) is actually one less than the largest negative integer (**Integer.MIN_VALUE**). If we add these two numbers together, the extra negative number will give us a result of **-1**.

Sample Multiple-Choice Question 3 Solution

03. (D)

Every location in **matrix** where the row index is equal to the column index is reassigned the value stored at index [0][0].

Sample Multiple-Choice Question 4 Solution

04. (C)

Method **papaya** computes the greatest common factor **(GCF)** of two integers.
Method **mango** uses the **GCF** from method **papaya** to compute the least common multiple **(LCM)** of the same two integers .

Sample Multiple-Choice Question 5 Solution

05. (A)

This program uses an **interface** and three implementing classes. Each class has a **greeting** method with a different implementation. This facilitates polymorphism. Choices **B** and **C** are not correct because the program displays complete sentences. Choices **D** and **E** are not correct because the program does compile and execute properly.

Sample Multiple-Choice Question 6 Solution

06. (C)

With double recursive call it is easier to do the computation backwards.

method2126(1) and method2126(2) == 1
method2126(3) == 3 + method2126(2) + method2126(1) = 3 + 1 + 1 == 5
method2126(4) == 4 + method2126(3) + method2126(2) = 4 + 5 + 1 == 10
method2126(5) == 5 + method2126(4) + method2126(3) = 5 + 10 + 5 == **20**

Sample Multiple-Choice Question 7 Solution

07. (E)

The program statement **x/[x.length]** uses **x.length** as an array index value.
This is greater than the largest index, which is always **length-1**.

Sample Multiple-Choice Question 8 Solution

08. (A)

The method gives the impression that it will create a copy of the *parameter string* with the vowels removed. The comparison statement is not done correctly. It should say:

if (!t.equals(“A”) && !t.equals(“E”) && !t.equals(“I”) && !t.equals(“O”) && !t.equals(“U”))

Since **!=** only compares the *shallow values*, the **if** statement will always have a **true** result. This means **temp**, which is being built one character at a time, will eventually store the exact same **String** value as **str**.

Sample Multiple-Choice Question 9 Solution

09. (B)

The temptation is to think that both **n1** and **n2** will be assigned random numbers in the range of **[0..1]**. What must be realized is that the **int** typecasting is applied to whatever value is returned by **Math.random()** before it can be multiplied by **2**. This means that both **n1** and **n2** will always will initialized with a value of **0**.

Sample Multiple-Choice Question 10 Solution

10. (B)

Consider the matrix below. The middle row shows the starting values in the **x** array at the start of executing **mystery1** and **mystery2**.

Method **mystery1** starts at the second number **2** and adds **1**.
Then number **3** + **3** becomes **6**, followed by number **4** + **6** becomes **10**, etc.

Method **mystery2** uses a different approach. The **increase** value is added to each number.
Each time in the loop **increase** increments by **1**. This creates the exact same sequence of numbers as **mystery1**.

1	3	6	10	15	21	28	36	45	55	66	78	91	105	120
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	3	6	10	15	21	28	36	45	55	66	78	91	105	120

Sample Free Response Question 1 Solution

Question 1.

Part (a).

```
public static String fromHexToBin(String hexNum)
{
    String binNum = "";
    for (int k = 0; k < hexNum.length(); k++)
    {
        String temp = hexNum.substring(k,k+1);
        if      (temp.equals("0")) binNum += "0000";
        else if (temp.equals("1")) binNum += "0001";
        else if (temp.equals("2")) binNum += "0010";
        else if (temp.equals("3")) binNum += "0011";
        else if (temp.equals("4")) binNum += "0100";
        else if (temp.equals("5")) binNum += "0101";
        else if (temp.equals("6")) binNum += "0110";
        else if (temp.equals("7")) binNum += "0111";
        else if (temp.equals("8")) binNum += "1000";
        else if (temp.equals("9")) binNum += "1001";
        else if (temp.equals("A")) binNum += "1010";
        else if (temp.equals("B")) binNum += "1011";
        else if (temp.equals("C")) binNum += "1100";
        else if (temp.equals("D")) binNum += "1101";
        else if (temp.equals("E")) binNum += "1110";
        else if (temp.equals("F")) binNum += "1111";
    }
    return binNum;
}
```

Part (b).

```
public static int fromBinToDec(String binNum)
{
    int decNum = 0;

    int pow2 = 1;
    for (int j = binNum.length()-1; j >= 0; j--)
    {
        if (binNum.charAt(j) == '1')
            decNum += pow2;
        pow2 *= 2;
    }
    return decNum;
}
```

Part (c).

```
public static int fromHexToDec(String binNum)
{
   String binNumber = fromHexToBin(binNum);
   int decNum = fromBinToDec(binNumber);
   return decNum;
}
```

Sample Free Response Question 2 Solution

Part (a).

```
public Pascal(int n)
{
    this.n = n;
    triangle = new int[n+1][];
    for (int degree = 0; degree < n+1; degree++)
    {
        triangle[degree] = new int[degree+1];
    }
}
```

Part (b).

```
public makeTriangle()
{
    for (int row = 0; row < triangle.length; row++)
    {
        for (int col = 0; col < triangle[row].length; col++)
        {
            if (col == 0 || col == triangle[row].length-1)
                triangle[row][col] = 1;
            else
                triangle[row][col] = triangle[row-1][col-1]
                                                + triangle[row-1][col];
        }
    }
}
```