1. Consider the following method:

```
public void
mysteryMix(Integer
a, Integer b)
{
     a = new
Integer(2 *
a.intValue());
     b = a;
}
```

What is the output of the following code?

```
Integer a = new
Integer(1);
Integer b = new
Integer(2);
mysteryMix(a, a);
mysteryMix(a, b);
System.out.println(a
+ " " + b);
```

a. 1 1
b. 1 2
c. 2 2
d. 1 4
e. 4 4

2.The following interface `Index` describes the location of a document:

```
public interface Index
{
     String getKey();
     Document
getAddress();
}
```

`Document` is a class that has a public method `getSize()`. An `ArrayList` `folder` contains `Index` objects and describes a collection of documents. Which of the following expressions refers to `size` of the `k`-th document in `folder`?

a. `folder[k-1].getAddress().getSize()`
b. `(Index) folder.get(k-1).getAddress().getSize()`
c. `((Index) folder.get(k-`

```
   1)).getAddress().getSize()
```
d. ((Document) (folder[k-1].getAddress())).getSize()

e. ((Document) (folder.get(k-1).getAddress())).getSize()

3. Which of the following could safely appear and make sense in place of < *condition* > in some suitable context?

```
     if ( <condition> )
          msg = new
message("o");
```

I. `msg == null || msg.getStatus().equals("x")`

II. `msg.getStatus().equals("x") || msg == null`

III. `"x".equals(msg.getStatus()) || msg == null`

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

4. What is the contents of the stack `s1` (its elements listed starting from the top) after the following code is executed?

```
Stack stk = new
ArrayStack();
Stack stk1 = new
ArrayStack();
Stack stk2 = new
ArrayStack();

int n;
Integer obj;
for (n = 1; n <= 6; n++)
     stk.push(new
Integer(n));

while (!stk.isEmpty())
{
     obj = (Integer
stk.pop());
     n = obj.intValue();
     if (N % 2 != 0)
          stk1.push(obj);
     else
          stk2.push(obj);
}
while (!stk1.isEmpty())
     stk.push(stk1.pop());
while (!stk2.isEmpty())
     stk.push(stk2.pop());
```

a. 1, 2, 3, 4, 5, 6
b. 6, 5, 4, 3, 2, 1
c. 1, 3, 5, 2, 4, 6
d. 2, 4, 6, 1, 3, 5
e. None of the above

5. Consider the following method:

```
public boolean someProperty(TreeNode root)
{
     return root != null &&
             (root.getLeft() != null &&
root.getRight() != null ||
                 someProperty(root.getLeft()) ||
                 someProperty(root.getRight()));
}
```

This method returns `true` if and only if the tree pointed to by `root`

a. is not empty.
b. is not empty and the root is not a leaf.
c. is not empty and the root is either a leaf or has two children.
d. has at least one node with two children.
e. is a full tree.

6. The method `max(TreeNode root)` assumes a precondition that `root` points to a non-empty binary search tree containing `Comparable` objects. `max` returns the value from the tree's largest node. Which of the following three versions of `max` return the correct answer when the precondition is met?

```
I.   public Object max(TreeNOde root)
     {
          while (root.getRight() != null)
               root = root.getRight();
          return root.getValue();
     }

II.  public Object max(TreeNOde root)
     {
          Object maxValue = root.getValue();
          if (root.getRight() != null)
          {
               Object temp = max(root.getRight());
               if
(((Comparable)temp).compareTo(maxValue) > 0)
                    maxValue = temp;
          }
          return maxValue;
     }
```

III. `public Object max(TreeNOde root)`

```
     {
          Object maxValue = root.getValue();

          if (root.getLeft() != null &&
               ((Comparable)max(root.getLeft())).
                    compareTo(maxValue) > 0)
               maxValue = max(root.getLeft());

          if (root.getRight() != null &&
               ((Comparable)max(root.getRight())).
                    compareTo(maxValue) > 0)
               maxValue = max(root.getRight());

          return maxValue;
     }
```

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

7. Given

```
int[] a = {1, 3, 5,
7, 9, 11, 13};
```

what are the values in a after `disarray(int[] a, int n)` is called? The method `disarray` is defined as follows:

```
public void
disarray(int[] a, int
n)
{
     if (n > 1)
     {
          disarray(a,
n-1);
          a[n-1] +=
a[n-2];
     }
}
```

a. 1, 4, 9, 16, 25, 36, 49
b. 1, 4, 8, 12, 16, 20, 24
c. 1, 8, 12, 16, 20, 24, 13
d. 1, 24, 20, 16, 12, 8, 4

e. None of the above

8. What is the value of n after the following code is executed?

```
int i, n = 0;
while (n < 90)
{
     for (i = 0; i <
10; i++)
     {
          n += 3;
          if (n > 50)
               break;
     }
     n++;
}
```

a. 51
b. 61
c. 91
d. 93
e. 104