1. The following program segment was proposed by a CS 61B student to print out whether or not a given year is a leap year. A leap year is either divisible by 400, or divisible by 4 but not by 100.

```java
if (year % 400 != 0)
    System.out.println(year + " is not a leap year");
else if (year % 100 != 0)
    System.out.println(year + " is a leap year");
else if (year % 4 != 0)
    System.out.println(year + " is not a leap year");
else
    System.out.println(year + " is a leap year");
```

For which values of `year` given below does the program correctly identify the year as a leap year or not a leap year? How would you fix the program?

1. 1904
2. 1900
3. 1905
4. 2000
2. Suppose I run the following code:

1. String a = "", b, c, d;
2. b = "I'm a string!";
3. c = b;
4. a = new String();
5. b = c;

How many variables, references, objects, and orphaned objects are there after each line of code? (An orphaned object is one which has no references pointing to it, which means that it can no longer be accessed in the program.)

<table>
<thead>
<tr>
<th>Variables</th>
<th>References</th>
<th>Objects</th>
<th>Orphaned objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>After line 1</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>After line 2</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>After line 3</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>After line 4</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td>After line 5</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

3. Suppose arrays were not built-in to Java, and you had to write an Array class. How would you declare the instance variable length? (For example, would it be private or public? What type of variable would it be? Would it be static?)

An example of a valid (but incorrect) answer:

```java
private static int length;
```
4. Write a method `selectionSort` that sorts an array of integers in place (that is, without creating a new array). It uses the following algorithm:

1. Consider the entire array as our subarray.
2. If the length of the subarray is \(\leq 1\), return.
3. Find the smallest item in the subarray.
4. Switch it with the first item in the subarray.
5. Set the subarray to be all but the first element of the subarray (since that element is in the right place). Go back to step 2.

*Make sure you do not create any new arrays!*

```java
public static void selectionSort(int[] nums) {

5. Consider the following classes:

```java
public class Mystery {
    public int theMystery = 42;
    public static void printTheMystery () {
        System.out.println(42);
    }
    public void printAgain () {
        System.out.println(theMystery);
    }
}
```
class Foo extends Mystery {
    public int theMystery = 13;
    public static void printTheMystery () {
        System.out.println(13);
    }
    public void printAgain () {
        System.out.println(theMystery);
    }
}

What will be printed out in the following cases? ("Compile-time error", "Run-time error" and "No output" are valid answers.)

> Mystery m = new Foo();

> System.out.println(m.theMystery);

> m.printTheMystery();

> m.printAgain();

> System.out.println(((Foo) m).theMystery);

> ((Foo) m).printTheMystery();

> ((Foo) m).printAgain();

> ((Mystery) m).printAgain();

> m = new Mystery();

> m.printAgain();

> ((Foo) m).printAgain();

> Foo f = new Mystery();

> Foo.printTheMystery();

> Foo.printAgain();