CPW 143 Fall 2015 Sample Final Exam

1. (12 points) Consider the following classes.

```
public class Vehicle { ... }
public class Car extends Vehicle {...}
public class LowRider extends Car {...}
```

Which of the following are legal statements? Circle the correct answers.

a) `Vehicle v = new Car();` **Legal** **Illegal**

b) `Vehicle v = new LowRider();` **Legal** **Illegal**

c) `Car c = new LowRider();` **Legal** **Illegal**

d) `Car c = new Vehicle();` **Legal** **Illegal**

e) `LowRider r = new LowRider();` **Legal** **Illegal**

f) `LowRider r = new Car();` **Legal** **Illegal**

2. (12 points) Write a method **`maxLength`** that takes an **`ArrayList`** of **`String`**s as a parameter and that returns the length of the longest **`String`** in the list. If your method is passed an empty **`ArrayList`**, it should return 0.

3. (12 points) Write the output that is printed when the given method below is passed each of the following maps as its parameter. Your answer should display the right values in the right order.

```
public static void mapMystery2(Map<String, String> m) {
    Set<String> s = new TreeSet<String>();
    for (String key : m.keySet()) {
        if (!m.get(key).equals(key)) {
            s.add(m.get(key));
        } else {
            s.remove(m.get(key));
        }
    }
    System.out.println(s);
}
```

A) `{sheep=wool, house=brick, cast=plaster, wool=wool}`
B) `{munchkin=blue, winkie=yellow, corn=yellow, grass=green, emerald=green}`
C) `{pumpkin=peach, corn=apple, apple=apple, pie=fruit, peach=peach}`
D) `{lab=ipl, lion=cat, terrier=dog, cat=cat, platypus=animal, nyan=cat}`

4. (12 points) Consider the following method:

```
public int mystery(int x, int y) {
    if (x > y) {
        return 0;
    } else {
        return mystery(x + 1, y) + 2 * x - 1;
    }
}
```

For each call below, indicate what value is returned:

Method Call	Value Returned
mystery(1, 3)	______________
mystery(4, 4)	______________
mystery(3, 5)	______________
mystery(1, 5)	______________
mystery(4, 7)	______________

5. (12 points) Write the output produced when the following method is passed each of the following stacks:

```
public static void mystery1(Stack<Integer> s) {
    Queue<Integer> q = new LinkedList<Integer>();
    while (!s.isEmpty()) {
        int n = s.pop();
        q.add(n);
        q.add(n);
    }
    while (!q.isEmpty()) {
        s.push(q.remove());
    }
    System.out.println(s);
}
```

A) `[2, 6, 1]`

B) `[30, 20, 10, 60, 50, 40]`

6. (12 points) Write a method **`starString`** that takes an integer *n* as a parameter and that returns a string of stars (asterisks) 2^n long (i.e., 2 to the *n*th power). For example:

   `starString(0)` should return `"*"` (because $2^0 = 1$)
   `starString(1)` should return `"**"` (because $2^1 = 2$)
   `starString(2)` should return `"****"` (because $2^2 = 4$)
   `starString(3)` should return `"********"` (because $2^3 = 8$)
   `starString(4)` should return `"****************"` (because $2^4 = 16$)

   Your method should take a single integer parameter that specifies the power of 2. You may assume the method will not be passed a value less than 0. You may NOT use a while loop, for loop or do/while loop to solve this problem; you MUST use recursion.

7. (12 points) Assume that the following classes have been defined.
   What output is produced by the following code?

```
public class Cass {
    public void method1() {
        System.out.print("cass 1  ");
    }

    public void method2() {
        System.out.print("cass 2  ");
    }

    public String toString() {
        return "cass";
    }
}

public class John extends Cass {
    public void method2() {
        method1();
        System.out.print("john 2  ");
    }

    public String toString() {
        return "john";
    }
}

public class Michelle extends John {
    public void method1() {
        System.out.print("michelle 1 ");
    }
}

public class Denny extends John {
    public void method1() {
        System.out.print("denny 1  ");
    }

    public String toString() {
        return "denny " +
                super.toString();
    }
}
```

```
Cass[] elements = { new Cass(),
                    new Denny(),
                    new John(),
                    new Michelle()};
for (int i = 0; i < elements.length;
i++) {
    elements[i].method1();
    System.out.println();
    elements[i].method2();
    System.out.println();
    System.out.println(elements[i]);
    System.out.println();
}
```

Output:

8. (12 points) Approximate the runtime of the following code fragment, in terms of **n**: Write your answer in a format such as "$O(n^2)$" or "$O(n \log n)$" (without the quotes).

```
int sum = 0;
for (int j = 1; j < n; j++) {
    sum += 5;
    if (j % 2 == 0) {
        sum++;
    }

}
```

9. (6 points) If you perform a linear search on an unsorted array of one million integers, looking for an integer that is not in the array, which of the following is the closest to the number of elements that the search algorithm will need to examine?
   a. All 1,000,000 of the integers
   b. Roughly 3/4 (750,000) of the integers
   c. Roughly 1/2 (500,000) of the integers
   d. Roughly 1/10 (100,000) of the integers
   e. Less than 1% (10,000 or fewer)

10. (6 points) If you perform a binary search on a sorted array of one million integers, looking for an integer that is not in the array, which of the following is the closest to the number of elements that the search algorithm will need to examine?
    a. All 1,000,000 of the integers
    b. Roughly 3/4 (750,000) of the integers
    c. Roughly 1/2 (500,000) of the integers
    d. Roughly 1/10 (100,000) of the integers
    e. Less than 1% (10,000 or fewer)