CS 145 Midterm #1 Practice Problems

Note that this is designed to give you a sample of the type of problems that may be on the exam, the length of the exam will be designed to be completed in one hour.

(1) **ArrayList**

Consider the following method:

```
public static void mystery2(ArrayList list) {
    for (int i = list.size() - 1; i > 0; i--) {
        if (list.get(i) < list.get(i - 1)) {
            int element = list.get(i);
            list.remove(i);
            list.add(0, element);
        }
    }
    System.out.println(list);
}
```

Write the output produced by the method when passed each of the following ArrayLists:

[2, 6, 1, 8]

[30, 20, 10, 60, 50, 40]

[-4, 16, 9, 1, 64, 25, 36, 4, 49]

(2) **ARRAY_LIST**

Write a method removeShorterStrings that takes an ArrayList of Strings as a parameter and that removes from each successive pair of values the shorter string in the pair. For example, suppose that an ArrayList called list contains the following values: {"four", "score", "and", "seven", "years", "ago"} In the first pair, "four" and "score", the shorter string is "four". In the second pair, "and" and "seven", the shorter string is "and". In the third pair, "years" and "ago", the shorter string is "ago". Therefore, the call: removeShorterStrings(list); should remove these shorter strings, leaving the list as follows: "score", "seven", "years". If there is a tie (both strings have the same length), your method should remove the first string in the pair. If there is an odd number of strings in the list, the final value should be kept in the list.

(3) **Classes**

```
public class Bay extends Lake {
    public void method1() {
        System.out.println("Bay 1");
        super.method2();
    }

    public void method2() {
        System.out.println("Bay 2");
    }
}

public class Pond {
    public void method2() {
        System.out.println("Pond 2");
    }
}

public class Ocean extends Bay {
    public void method2() {
        System.out.println("Ocean 2");
    }
}

public class Lake extends Pond {
    public void method3() {
        System.out.println("Lake 3");
        method2();
    }
}
```

Suppose the following variables are defined:

```
Lake var1 = new Ocean();
Pond var2 = new Pond();
Pond var3 = new Lake();
Object var4 = new Bay();
Lake var5 = new Bay();
Bay var6 = new Ocean();
```

What would the output of the following method calls be?

```
var1.method2();       ((Ocean) var5).method1();
var2.method2();       ((Lake) var3).method3();
var3.method2();       ((Lake) var4).method1();
var4.method2();       ((Ocean) var1).method1();
var5.method2();       ((Bay) var4).method1();
var6.method2();       ((Lake) var2).method3();
var1.method3();       ((Ocean) var5).method1();
var2.method3();       ((Pond) var4).method2();
var3.method3();       ((Bay) var4).method1();
var4.method3();       ((Lake) var2).method3();
var5.method3();       ((Ocean) var5).method1();
var6.method3();       ((Pond) var4).method2();
```

(4) **Sets**

Write a method removeEvenLength that takes a Set of strings as a parameter and that removes all of the strings of even length from the set. For example, if your method is passed a set containing the following elements:

["foo", "buzz", "bar", "fork", "bort", "spoon", "!", "dude"]
Your method should modify the set to store the following elements (the order of the elements does not matter):

["foo", "bar", "spoon", "!"]

(5) **Collections Programming.**

Write a method countInAreaCode that accepts two parameters, a Map from names (strings) to phone numbers (strings) and an area code (as a string), and returns how many unique phone numbers in the map use that area code. For example, if a map m contains these pairs:

{Marty=206-685-2181, Rick=520-206-6126, Beekto=206-685-2181,
Jenny=253-867-5309, Stuart=206-685-9138, DirecTV=800-494-4388,
Bob=206-685-9138, Benson=206-616-1246, Hottline=900-520-2767}

The call of countInAreaCode(m, "206") should return 3, because there are 3 unique phone numbers that use the 206 area code: Marty/Beekto's number of "206-685-2181", Stuart/Bob's number of "206-685-9138", and Benson's number of "206-616-1246".

You may assume that the map passed is not null, that no key or value in it is null, that every phone number value string in the map will begin with

a 3-digit numeric area code, and that the area code string passed will be a non-null numeric string exactly 3 characters in length. If the map is empty or contains no
phone numbers with the given area code, your method should return 0.

You may create one collection of your choice as auxiliary storage to solve this problem. You can have as many simple variables as you like. You should not modify the contents of the map passed to your method.

(6) **Recursion**

Write a recursive method digitSum() that finds the sum of the digits of an integers.   For example digitSum(1234) would return 10 because 1+2+3+4=10.   You could do this with a for loop, but can you do it recursively.

(7)   **Recursive tracing**

Trace through the following recursive method given the outputs below:

```
public static int foo(int x, int y)
{

if (x <0 || y < 0) return -1;
if (y == 0) return 0;

return x + foo(y-1, x)
}

foo(5, 7);
foo(6, 1);
```

(8) **Recursion**

Write a recursive method moveToEnd that accepts a String s and a char c as parameters, and returns a new String similar to s but with all occurrences of c moved to the end of the string. The relative order of the other characters should be unchanged from their order in the original string s. If s does not contain c, it should be returned unmodified.

The following table shows calls to your method and their return values.

Call	Value Returned
moveToEnd("hello", 'l')	"heoll"
moveToEnd("hello", 'e')	"hlloe"
moveToEnd("hello there", 'e')	"hllo threee"
moveToEnd("hello there", 'q')	"hello there"
moveToEnd("HELLO there", 'e')	"HELLO three"
moveToEnd("", 'x')	""

(9) **RECURSIVE TRACE**

For each call to the following method, indicate what value is returned:

```
      public int bar(int x, int y) {
          if (x < 0) {
              return -bar(-x, y);
          } else if (y < 0) {
              return -bar(x, -y);
          } else if (x == 0 && y == 0) {
              return 0;
          }

          return 100 * bar(x / 10, y / 10) + 10 * (x % 10) + y % 10;
    }
}
```

- bar (5, 7)
- bar (12, 9)
- bar (-7, 4)
- bar (-23, -48)
- bar (128, 343)