CSE142 Sample Final Exam
Spring 2018

1. Reference Mystery, 5 points. Consider the following class declaration:

```
public static class BasicPoint {
    int x;
    int y;

    public BasicPoint(int initialX, int initialY) {
        x = initialX;
        y = initialY;
    }
}
```

Given the previous declaration, what output would be produced by the following program?

```
import java.util.*;

public class ReferenceMystery {
    public static void main(String[] args) {
        BasicPoint p = new BasicPoint(11, 22);
        int[] a = {33, 44};
        int n = 55;

        mystery1(p, a, n);
        System.out.println(p.x + "," + p.y + " " + Arrays.toString(a) + " " + n);

        a[0] = a[1];
        p.x = p.y;

        n = mystery2(a, n);
        System.out.println(p.x + "," + p.y + " " + n);
    }

    public static int mystery1(BasicPoint p, int[] a, int n) {
        n = 0;
        a[0] = a[0] + 11;
        a[1] = 77;
        p.x = p.x + 33;
        System.out.println(p.x + "," + p.y + " " + Arrays.toString(a) + " " + n);
        return n;
    }

    public static int mystery2(int[] a, int n) {
        n = a[0];
        a[0] = a[0] + 11;
        a[1] = n + 11;
        return n;
    }
}
```

2. Array Simulation, 10 points. You are to simulate the execution of a method that manipulates an array of integers. Consider the following method:

```
public static void mystery(int[] a) {
   for (int i = 1; i < a.length - 1; i++) {
     a[i] = (a[i - 1] + a[i + 1]) / 2;
   }
}
```

In the left-hand column below are specific arrays of integers. You are to indicate in the right-hand column what values would be stored in the array after method mystery executes if the integer array in the left-hand column is passed as a parameter to mystery.

Original Array	Final Array
{1, 1, 3}	________________________________
{2, 1, 2, 4}	________________________________
{6, 13, 0, 3, 7}	________________________________
{-1, 6, 3, 5, -3}	________________________________
{7, 2, 3, 1, -3, 12}	________________________________

3. Inheritance, 6 points. Assume that the following four classes have been defined:

```
public class Diamond extends Ruby {
    public String toString() {
        return "Diamond";
    }
}

public class Garnet extends Diamond {
    public String toString() {
        return "Garnet";
    }

    public void method1() {
        System.out.println("Garnet 1");
    }
}

public class Ruby extends Emerald {
    public void method1() {
        System.out.println("Ruby 1");
    }
}

public class Emerald {
    public String toString() {
        return "Emerald";
    }

    public void method1() {
        System.out.println("Emerald 1");
    }

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

And consider the following code fragment:

```
Emerald[] gems = { new Diamond(), new Ruby(),
                   new Emerald(), new Garnet() };
for (int i = 0; i < gems.length; i++) {
    System.out.println(gems[i]);
    gems[i].method1();
    gems[i].method2();
    System.out.println();
}
```

In the space to the right, write the output that would be produced by this code exactly as it would appear on the console.

4. Token-Based File Processing, 10 points. Write a static method named censorNames that accepts one parameter: a Scanner for an input file containing the first and last names of several people. Your method should print each person's first name followed by a censored version of their last name. Your method should also return the total number of people named in the file.

   The input file contains a series of pairs of first names and last names. The input may span multiple lines and may have different spacing between tokens. You may assume that each first name will be followed by a last name.

   Your method should print one full name per line. Each person's first name will simply be printed as it appears in the file. Instead of printing last names, print a series of 'X' characters of the same length as the original last name. The first and last names should be separated by a space.

   For example, given a Scanner named input referring to an input file that contains the following data:

```
Whitaker Brand Malcolm X Grace Hopper
         Alan          Turing STUART
   Reges
```

   If we made the following call:

```
censorNames(input)
```

   we would expect the following output:

```
Whitaker XXXXX
Malcolm X
Grace XXXXXX
Alan XXXXXX
STUART XXXXX
```

   This call would return the value 5. You may assume that the input file exists and has the format described above. The file will always contain at least one person's first and last names and will always contain an even number of tokens.

5. Line-Based File Processing, 9 points. Write a static method called analyzeParagraphs that takes as a parameter a Scanner containing a text file and that produces output that describes the paragraph structure of the file, returning the maximum number of lines in any given paragraph. Each paragraph in the input file will be terminated by the text "<p>" on a line by itself.

   For example, consider the following input file:

```
This is an example of an input file
with four different paragraphs.
<p>
The second paragraph is the longest
with three lines, so your method should
return 3 when processing this file.
<p>
<p>
The third paragraph was empty.  This one is just short.
<p>
```

   The method should count the number of lines in each paragraph and report that information to System.out. For example, if the input above is stored in a Scanner called input and we make the following call:

```
analyzeParagraphs(input);
```

   we would expect the following output:

```
2-line paragraph
3-line paragraph
0-line paragraph
1-line paragraph
```

   This call would return the value 3, the maximum number of lines in any given paragraph. You must exactly reproduce the format of this output. You may assume that the input file has no blank lines, that it contains at least one paragraph, and that each paragraph is terminated by a line containing just "<p>".

6. Arrays, 10 points. This problem is a variation of the DNA analysis program you wrote in assignment 7. In this version of the problem, we will imagine that each individual nucleotide in a sequence has been assigned a value representing how important that nucleotide is to our analysis. You will write code to compute the total value of each of the four nucleotides (A, C, G, T) in a given sequence. (Note that this variation is not necessarily based on any real-life science.)

   Write a static method called getTotalValues that takes a String and an array of doubles as parameters. The array will contain the same number of elements as the number of non-junk nucleotides in the string, and each element of the array contains the value of the corresponding non-junk nucleotide in the string. (That is, the first element of the array corresponds to the first non-junk nucleotide, the second element to the second non-junk nucleotide, and so on. See below for a more detailed example.) Your method should return an array containing the total values of A, C, G, and T (in that order) represented by the parameters.

   For example, suppose the following declarations are made:

```
String seq = "GA-CAAC-G--C";
double[] vals = {1.0, 2.1, 1.3, 0.7, 3.4, 2.0, 1.0, 0.6};
```

   In this case, the first nucleotide, a G, is considered to have value 1.0. The second nucleotide, a A, has a value of 2.1; the third non-junk nucleotide, a C, has a value of 1.3; and so on.

   Suppose the following call is then made:

```
double[] totals = getTotalValues(seq, vals);
```

   After the call, totals would contain:

```
[6.2, 3.9, 2.0, 0.0]
```

   Where 6.2 is the total value of all A's in the sequence, 3.9 is the total value of all C's, 2.0 is the total value of all G's, and 0.0 is the total value of all T's (since there were no T's in the input string).

   You may assume that number of elements in the array and the number of non-junk nucleotides in the string are always equal. You may also assume that each character in the string parameter will be one of 'A', 'C', 'G', 'T', or '-' (letters will all be uppercase) and that each value in the array is nonnegative.

7. ArrayList, 10 points. Write a static method called split that takes an ArrayList of integer values as a parameter and that replaces each value in the list with a pair of values, each half the original. If a number in the original list is odd, then the first number in the new pair should be one higher than the second so that the sum equals the original number. For example, if a variable called list stores this sequence of values:

```
[18, 7, 4, 24, 11]
```

The number 18 is split into the pair (9, 9), the number 7 is split into (4, 3), the number 4 is split into (2, 2), the number 24 is split into (12, 12) and the number 11 is split into (6, 5). Thus, the call:

```
split(list);
```

should cause list to store the following sequence of values afterwards:

```
[9, 9, 4, 3, 2, 2, 12, 12, 6, 5]
```

You may assume that all numbers in the list are nonnegative. You may only use ArrayList methods listed on the cheat sheet. You may not construct any extra data structures or String objects to solve this problem. You must solve it by manipulating the ArrayList you are passed as a parameter.

8. Critters, 15 points. Write a critter class called Panther along with its movement, fighting, eating, and appearance. All unspecified aspects of Panther use the default Critter behavior. Write the complete class with any fields, constructors, etc. necessary to implement the behavior.

Panthers move randomly around the world, but never stand still. That is, a Panther is equally likely to move north, south, east, or west on any given move, but will never fail to move (i.e. Direction.CENTER).

A Panther is always in one of two modes: foraging or hunting. Panthers are in foraging mode when initially created. While foraging, a Panther should display as black and should always roar when fighting. A Panther that is foraging continues foraging until it finds food.

When a foraging Panther encounters food, the Panther should eat it, and then switch to hunting mode. Hunting Panthers should display as red and should always scratch when fighting. A hunting Panther continues hunting until it gets into a fight, at which point it returns to foraging mode. Hunting Panthers do not eat.

9. Arrays, 15 points. Write a static method named insertMiddle that accepts two arrays of integers, a and b, as parameters and returns a new array containing elements from the first half of a followed by all the elements of b followed by elements from the second half of a. For example, consider the following two arrays:

```
int[] a = {2, 4, 6, 8, 10};
int[] b = {1, 1, 1};
```

The call insertMiddle(a, b); should return the following array:

```
{2, 4, 1, 1, 1, 6, 8, 10}
```

Notice that if a has an odd length, its shorter half goes first.

You may not construct any extra data structures or String objects to solve this problem; you may only create the one array that you are to return. You may not modify the arrays that are passed in.

10. Programming, 10 points. Write a static method called samePattern that returns true or false depending upon whether two strings have the same pattern of characters. More precisely, two strings have the same pattern if they are of the same length and if two characters in the first string are equal if and only if the characters in the corresponding positions in the second string are also equal. Below are some examples of patterns that are the same and patterns that differ (keep in mind that the method should return the same value no matter what order the two strings are passed).

1st String	2nd String	Same Pattern?
""	""	true
"a"	"x"	true
"a"	"ab"	false
"ab"	"ab"	true
"aa"	"xy"	false
"aba"	"+-+"	true
"---"	"aba"	false
"abcabc"	"zodzod"	true
"abcabd"	"zodzoe"	true
"abcabc"	"xxxxxx"	false
"aaasssccn"	"aaabbbcccd"	true
"asasasasas"	"xyxyxyxyxy"	true
"ascneencsa"	"aeiouuoiea"	true
"aaasssccn"	"aaabbbcccd"	true
"asasasasas"	"xxxxxyyyyy"	false
"ascneencsa"	"aeiouaeiou"	false
"aaasssccn"	"xxxyyyzzzz"	false
"aaasssiiii"	"gggdddfffh"	false

Your method should take two parameters: the two strings to compare. You are allowed to create new strings, but otherwise you are not allowed to construct extra data structures to solve this problem (no array, ArrayList, Scanner, etc). You are limited to the String methods on the cheat sheet.