CSE142 Sample Final Exam, Winter 2018

1. Reference Mystery, 5 points. What output is produced by this program?

```
import java.util.*;

public class Rectangle {
    int w;
    int h;

    public Rectangle(int width, int height) {
        w = width;
        h = height;
    }

    public String toString() {
        return "w: " + w + ", h: " + h;
    }
}

public class ReferenceMystery {
    public static void main(String[] args) {
        int n = 20;
        int[] a = {40};  // an array with just one element
        Rectangle r = new Rectangle(50, 10);

        mystery(n, a, r);
        System.out.println(n + " " + Arrays.toString(a) + " " + r);

        a[0]++;
        r.w++;
        mystery(n, a, r);
        System.out.println(n + " " + Arrays.toString(a) + " " + r);
    }

    public static int mystery(int n, int[] a, Rectangle r) {
        n++;
        a[0]++;
        r.h++;
        System.out.println(n + " " + Arrays.toString(a) + " " + r);
        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[] list) {
    for (int i = 1; i < list.length - 1; i++) {
        list[i] = list[i - 1] + list[i + 1];
    }
}
```

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

Original List	Final List
{2, 5, 6, 9}	_____
{1, 3, 3, 5, 8}	_____
{4, 4, 5, 8, 2}	_____
{3, 6, 9, 12, 15}	_____
{2, 4, 6, 8, 10, 12}	_____

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

```
public class C extends B {
    public void method2() {
        System.out.println("c 2");
    }
}

public class B {
    public String toString() {
        return "b";
    }

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

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

public class A extends D {
    public void method1() {
        System.out.println("a 1");
    }

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

public class D extends C {
    public String toString() {
        return "d";
    }
}
```

Consider the following code fragment:

```
B[] elements = {new C(), new A(), new D(), new B()};
for (int i = 0; i < elements.length; i++) {
    System.out.println(elements[i]);
    elements[i].method1();
    elements[i].method2();
    System.out.println();
}
```

What output is produced by this code? Write the output as a series of 3-line columns in order from left to right (do not label columns or rows).

4. Token-Based File Processing, 10 points. Write a static method called redact that takes as a parameter a Scanner containing a text with special markers indicating sensitive words that are to be replaced. It prints the resulting text with each word on a separate line. The idea is that the text contains potentially classified information and extra "content markers" have been included throughout to indicate classified material. The special string "CLASSIFIED" appears in various parts of the text to indicate sensitive material. Each occurrence of the string will be followed by an integer indicating how many words are to be redacted. For each of those words, you should print the text "[redacted]" instead of printing the word. For example, if a Scanner called text contains the following:

```
four score CLASSIFIED 3 and seven years ago our CLASSIFIED 1 fathers
brought forth CLASSIFIED 2 on this continent
```

There are three indications of classified material. If you were to call the method as follows:

```
redact(text);
```

the following output should be produced:

```
four
score
[redacted]
[redacted]
[redacted]
ago
our
[redacted]
brought
forth
[redacted]
[redacted]
continent
```

You are to exactly reproduce the format of this output. Notice that line breaks in the input are not meaningful. You may not construct any extra data structures to solve this problem.

5. Line-Based File Processing, 9 points. Write a static method called diff that takes two Scanners as parameters and that reports any differences between them. Assume that both files have the same number of lines. Your method should ignore lowercase versus uppercase letters, but should report each pair of lines that differ in other ways. For example, suppose that scanners called input1 and input2 contain the following lines of text:

```
      input1                        input2
------------------            ------------------
this is an example            This is an EXAMPLE
Input File With               input feel with
some differences              some diFFerences
but many lines                BuT mAnY lines
the same!                     The same.
```

If you make the call:

```
diff(input1, input2);
```

the following output should be produced:

```
DIFF: difference found on line #2
[Input File With] versus [input feel with]
DIFF: difference found on line #5
[the same!] versus [The same.]

2 line(s) are different
```

For each pair of lines that differ, the output lists the line number on a first line of output followed by a second line of output that shows the text from the two different files. It also reports the total number of lines that differ at the end. You are to exactly reproduce the format of this output. Also notice that differences in case don't matter. The first reported difference involves different words ("File" versus "feel"). The second reported difference involves an exclamation mark versus a period.

If no differences are found, then your method should produce no output (not even the report of how many differences were found). You may not construct any extra data structures to solve this problem other than strings.

6. Arrays, 10 points. Write a static method called switchPairs that switches the order of elements in an array of integers in a pairwise fashion. Your method should switch the order of the first two values, then switch the order of the next two, switch the order of the next two, and so on. For example, suppose that a variable called list stores the following:

```
[12, 4, 8, 7, 9, -3]
```

This list has three pairs: (12, 4), (8, 7), and (9, -3). Thus, the call:

```
switchPairs(list);
```

should leave the list with these values:

```
[4, 12, 7, 8, -3, 9]
```

Notice that each pair has been switched. If there are an odd number of values in the list, the final element should not be moved. For example, if the original list had been:

```
[12, 4, 8, 7, 9, -3, 42]
```

It would again switch pairs of values, but the final value (42) would not be moved, yielding this list:

```
[4, 12, 7, 8, -3, 9, 42]
```

You may not construct any extra data structures to solve this problem.

7. ArrayList, 10 points. Write a static method called negativesToFront that takes an ArrayList of integer values as a parameter and that moves all negative numbers to the front of the list, preserving their relative order. For example, if a variable called list stores this sequence of values:

```
[4, -5, 3, 6, -2, -9, 14, -3, -10, 42, 18]
```

then the following call:

```
negativesToFront(list);
```

should leave the list with the following values:

```
[-5, -2, -9, -3, -10, 4, 3, 6, 14, 42, 18]
```

Notice that the list begins with the negatives values in their original order followed by the nonnegative values in their original order. You may not construct any extra data structures to solve this problem. You must solve it by manipulating the ArrayList you are passed as a parameter. See the cheat sheet for a list of available ArrayList methods.

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

   A WatchDog walks back and forth, pausing at either end of its territory. The WatchDog is well trained, so it ignores any food it comes across while on patrol. When it fights, it always roars. (er, barks.) It normally moves 5 paces west, then pauses for 1 move (Direction.CENTER), then moves 5 paces east, then pauses for 1 move (spaces inserted for readability):

   ```
   WWWWW C EEEEE C
   ```

   However, WatchDogs are morbidly afraid of Ants: if a WatchDog ever fights an Ant and lives, then the WatchDog should go into a panic "flight" mode: run in a direction, and keeping running that way. The WatchDog constructor takes a Direction parameter (called, say, panicDirection) which represents the direction that the WatchDog should run while it is in panic mode.

   While the WatchDog is in a panic, it will choose to eat any food that it encounters. If it does eat food, then it should regain its composure and begin patrolling again: panic mode comes to an end. It should continue where it left off in its patrol cycle, even though the WatchDog may be in an entirely different part of the critter world.

   If you constructed a WatchDog critter, and then after 2 moves it fights an Ant, the WatchDog should run until it finds (and eats) a food, and then resume where it left off in its patrol cycle:

   ```
   Critter watchDog = new WatchDog(Direction.SOUTH);

   __% <- fights ant_______eats food-> .____________________
   WW SSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS WWWCEEEEECWWW(....)
   ```

   The WatchDog should be represented with a D usually. When in a panic, its string should be an exclamation point: !

9. Arrays, 15 points. Write a static method called expand that takes an array of count/value pairs and that constructs and returns a new array containing the expansion of those count/value pairs. For example, suppose that a variable list has been defined as follows (pairs indicated for emphasis):

```
int[] list = {3, 8, 4, 2, 0, 42, 5, 1};
              |  |  |  |  |   |  |  |
              +--+  +--+  +---+  +--+
              pair  pair  pair   pair
```

This array indicates that the result should have 3 occurrences of 8 followed by 4 occurrences of 2 followed by 0 occurrences of 42 followed by 5 occurrences of 1. So the call expand(list) would construct and return the following array:

```
{8, 8, 8, 2, 2, 2, 2, 1, 1, 1, 1, 1}
```

Solving this problem will require making two passes through the original array. In the first pass, your method should compute the overall length required for the new array. In the second pass, it should fill up the new array to be returned. You may assume that the array passed to your method has a legal sequence of count/value pairs with no negative counts. Notice, however, that a count might be 0, as in the example above.

You may not construct any String objects or extra data structures other than the array that is returned to solve this problem. You may not modify the array that is passed in.

10. Programming, 10 points. Write a static method called compress that takes an ArrayList of strings as a parameter and that replaces each sequence of two or more equal strings in the list with a single string consisting of the number of strings that were equal, an asterisk, and the original string. For example, suppose that an ArrayList called list contains the following:

```
["clam", "squid", "squid", "squid", "clam", "octopus", "octopus"]
```

if we make the call:

```
compress(list);
```

then after the call the list should store the following values:

```
["clam", "3*squid", "clam", "2*octopus"]
```

Notice that the 3 occurrences of "squid" have been replaced with "3*squid", and the 2 occurrences of "octopus" have been replaced with "2*octopus". The two occurrences of "clam" haven't been changed, since they weren't next to each other.

You are not allowed to construct extra data structures to solve this problem (no array, ArrayList, Scanner, String, etc).