CSE142 Section Handout #9 Questions

1. Expressions, 5 points. For each expression in the left-hand column, indicate its value in the right-hand column. Be sure to list a constant of appropriate type (e.g., 7.0 rather than 7 for a double, Strings in quotes).

Expression	Value
(6 + 7) / 4 / 2.0	__________
13 / 2 * 3 % 10 - 1	__________
30 / (2 * 6) + 1.5	__________
13 % 5 + 5 * 3 / 4	__________
"3 * 4" + 3 * 4 + 10	__________

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; i++) {
        if (list[i - 1] % 2 == 0) {
            list[i - 1]++;
            list[i]++;
        }
    }
}
```

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
[12, 7]	____________________________
[2, 3, 4, 5, 6]	____________________________
[3, 4, 5, 7, 9]	____________________________
[2, 3, 5, 7, 9]	____________________________
[4, 5, 9, 6, 2]	____________________________

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

```
public class A extends B {
    public String toString() {
        return "a";
    }
}

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

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

```
public class C extends D {
    public void method1() {
        System.out.println("c 1");
    }
}

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

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

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

Consider the following code fragment:

```
D[] elements = {new B(), new A(), new D(), new C()};
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? (you may write the output as a series of 3-line columns in order from left to right)

4. Token-Based File Processing, 10 points. Write a static method called showSums that takes as a parameter a Scanner containing a sequence of integers and that reports each cumulative sum of the sequence and the average of the numbers. For example, if the following text is stored in a Scanner called data:

```
8 4 13 5 9
```

and we make the following call:

```
showSums(data);
```

your method should produce the following output:

```
Sum of 1 = 8
Sum of 2 = 12
Sum of 3 = 25
Sum of 4 = 30
Sum of 5 = 39
Average = 7.8
```

Notice that the various lines of output report the sum including just the first number, then including the first two numbers, then including the first three numbers, and so on, up to the sum including all numbers. The final line of output reports the average of the sequence. Notice that this is the average of the numbers themselves, not the average of the cumulative sums.

If the Scanner contains the following values:

```
1 2 3 4
```

the method should produce the following output:

```
Sum of 1 = 1
Sum of 2 = 3
Sum of 3 = 6
Sum of 4 = 10
Average = 2.5
```

You are to exactly reproduce the format of these sample outputs. You may assume that the Scanner has at least one integer to be processed.

5. Line-Based File Processing, 9 points. Write a static method called underline that takes a Scanner containing an input file as a parameter and that prints to System.out the same text with certain lines underlined. The lines to be underlined all begin with a period. The period should not be printed. You should print the text that follows the period on a line by itself followed by a line of dashes equal in length to the text that follows the period. For example, consider the following input:

```
.Statement of Purpose
I didn't expect to major in computer science until I took cse142.
I liked it more than I expected and that got me hooked on cs.

.High School Performance
I got very good grades in high school, graduating in the top 10% of
my class.

.College Performance
I have done well in my college classes, with an overall gpa of 3.5.
```

If the text above is stored in a Scanner called input and we make this call:

```
underline(input);
```

the method should print the following output to System.out:

```
Statement of Purpose
--------------------
I didn't expect to major in computer science until I took cse142.
I liked it more than I expected and that got me hooked on cs.

High School Performance
-----------------------
I got very good grades in high school, graduating in the top 10% of
my class.

College Performance
-------------------
I have done well in my college classes, with an overall gpa of 3.5.
```

Notice that some of the input lines can be blank lines.

6. Arrays, 10 points. Write a static method called hasAlternatingParity that returns whether or not an array of integers has alternating parity (true if it does, false otherwise). The parity of an integer is 0 for even numbers and 1 for odd numbers. To have alternating parity, a list would have to alternate between even and odd numbers, as in the following list:

```
[3, 2, 19, 8, 43, 64, 1, 0, 3]
```

If these values are stored in an array called data and we make this call:

```
hasAlternatingParity(data)
```

the method would return true. If the array instead stored these values:

```
[2, 13, 4, 1, 0, 9, 2, 7, 4, 12, 3, 2]
```

it would return false because there are two even numbers in a row (4, 12).

By definition, an empty list or a list of one element has alternating parity. You may assume the values in the array are not negative.

7. ArrayList, 10 points. Write a static method called switchPairs that switches the order of values in an ArrayList of Strings in a pairwise fashion. It 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, if a variable called list stores these values:

```
["four", "score", "and", "seven", "years", "ago"]
```

and we make this call:

```
switchPairs(list);
```

the method should switch the first pair ("four", "score"), the second pair ("and", "seven") and the third pair ("years", "ago"), yielding this list:

```
["score", "four", "seven", "and", "ago", "years"]
```

If there are an odd number of values in the list, the final element is not moved. For example, if the original list had been:

```
["to", "be", "or", "not", "to", "be", "hamlet"]
```

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

```
["be", "to", "not", "or", "be", "to", "hamlet"]
```

8. Critters, 15 points. Write a class called Chameleon that extends the Critter class. The instances of the Chameleon class cycle through three different ways of displaying themselves. On their first move they should appear as a red R. On their second move they should appear as a white W. On their third move they should appear as a blue B. Then this pattern repeats itself (red R, white W, blue B, red R, white W, blue B, etc). They should always infect on moves when they are red no matter what is in front of them. On moves when they are white or blue, they should hop when they can and turn right when they can't hop.

9. Arrays, 15 points. Write a static method called doubleSize that takes an array of integers as an argument and that returns a new array twice as large as the original that replaces every integer from the original list with a pair of integers, 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 stretched into (6, 5). Thus, the call:

```
int[] result = doubleSize(list);
```

should return an array containing this sequence:

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

Your method should not change the array passed as a parameter. You are not allowed to solve this problem using an ArrayList or Scanner and you are not allowed to call any methods of the Arrays class or the Collections class.

10. Programming, 10 points. Write a static method called hasTwoPair that takes an array of integers in the range of 1 to 6 as a parameter and that returns whether or not the array contains two values that appear twice (true if it does, false if it does not). This is a problem from the game of Yahtzee in which players roll five dice and look for various combinations, but your solution should not depend on the array containing five values. You can, however, make use of the fact that all of the numbers will be in the range of 1 to 6.

Your method should return true when exactly two values appearing in the array occur exactly two times. If one value appears two times and another appears three times, that would not count as two pairs. Similarly, if there are three numbers that appear two times, that would not count as two pairs. There must be exactly two such numbers and each must occur exactly two times. Below are examples of arrays and the value that should be returned for each by hasTwoPair:

Contents of array	Value returned
{2, 4, 2, 2, 4}	false
{3, 4, 3, 6, 6}	true
{4, 1, 4, 4, 2}	false
{5, 5, 3, 3, 4}	true
{6, 2, 6, 5, 3}	false
{1, 3, 5, 3, 1}	true
{3, 1, 3, 1}	true
{1, 2, 3, 1, 2, 3}	false

CSE142 Section Handout #9 Solutions

1.

Expression	Value
(6 + 7) / 4 / 2.0	1.5
13 / 2 * 3 % 10 - 1	7
30 / (2 * 6) + 1.5	3.5
13 % 5 + 5 * 3 / 4	6
"3 * 4" + 3 * 4 + 10	"3 * 41210"

2.

Original List	Final List
[12, 7]	[13, 8]
[2, 3, 4, 5, 6]	[3, 5, 5, 5, 6]
[3, 4, 5, 7, 9]	[3, 5, 7, 9, 10]
[2, 3, 5, 7, 9]	[3, 5, 7, 9, 10]
[4, 5, 9, 6, 2]	[5, 7, 11, 7, 2]

3. Inheritance. The output produced is as follows.

```
d        a        d        d
b 1      b 1      d 1      c 1
b 2      b 2      d 2      d 2
```

4. Token-Based File Processing. One possible solution appears below.

```
public static void showSums(Scanner input) {
    int count = 0;
    int sum = 0;
    while (input.hasNextInt()) {
        sum += input.nextInt();
        count++;
        System.out.println("Sum of " + count + " = " + sum);
    }
    double average = (double) sum / count;
    System.out.println("Average = " + average);
}
```

5. Line-Based File Processing. One possible solution appears below.

```
public static void underline(Scanner input) {
    while (input.hasNextLine()) {
        String text = input.nextLine();
        if (!text.startsWith(".")) {
            System.out.println(text);
        } else {
            System.out.println(text.substring(1));
            for (int i = 1; i <= text.length() - 1; i++) {
                System.out.print("-");
            }
            System.out.println();
        }
    }
}
```

6. Arrays. One possible solution appears below.

```
        public static boolean hasAlternatingParity(int[] list) {
            for (int i = 0; i < list.length - 1; i++) {
                if (list[i] % 2 == list[i + 1] % 2) {
                    return false;
                }
            }
            return true;
        }
```

7. ArrayLists. Two possible solutions appear below.

```
        public static void switchPairs(ArrayList<String> list) {
            for (int i = 0; i < list.size() - 1; i += 2) {
                String first = list.remove(i);
                list.add(i + 1, first);
            }
        }

        public static void switchPairs(ArrayList<String> list) {
            int i = 0;
            while (i < list.size() - 1) {
                String first = list.get(i);
                list.set(i, list.get(i + 1));
                list.set(i + 1, first);
                i += 2;
            }
        }
```

8. Critters. One possible solution appears below.

```
        public class Chameleon extends Critter {
            private int count = 0;

            public Action getMove(CritterInfo info) {
                count++;
                if (count % 3 == 1) {
                    return Action.INFECT;
                } else if (info.getFront() == Neighbor.EMPTY) {
                    return Action.HOP;
                } else {
                    return Action.RIGHT;
                }
            }

            public Color getColor() {
                if (count % 3 == 0) {
                    return Color.RED;
                } else if (count % 3 == 1) {
                    return Color.WHITE;
                } else {
                    return Color.BLUE;
                }
            }
            public String toString() {
                if (count % 3 == 0) {
                    return "R";
                } else if (count % 3 == 1) {
                    return "W";
                } else {
                    return "B";
                }
            }
        }
```

9. Arrays. One possible solution appears below.

```
        public static int[] doubleSize(int[] list) {
            int[] result = new int[2 * list.length];
            for (int i = 0; i < list.length; i++) {
                result[2 * i] = list[i] / 2 + list[i] % 2;
                result[2 * i + 1] = list[i] / 2;
            }
            return result;
        }
```

10. Programming. One possible solution appears below.

```
        public static boolean hasTwoPair(int[] data) {
            int[] counts = new int[6];
            for (int i = 0; i < data.length; i++) {
                counts[data[i] - 1]++;
            }
            int num = 0;
            for (int i = 0; i < counts.length; i++) {
                if (counts[i] == 2) {
                    num++;
                }
            }
            return num == 2;
        }
```