CSE142 Section Handout #9 Questions

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

```
import java.util.*;
public class ReferenceMystery {
    public static void main(String[] args) {
        int x = 4;
        int y = 8;
        int[] data = {5, 10, 15};
        x = mystery1(y, data);
        System.out.println(y + " " + Arrays.toString(data));
        mystery2(x, y);
        System.out.println(x + " " + y);
    }

    public static int mystery1(int n, int[] numbers) {
        n = n + 100;
        numbers[2]--;
        System.out.println(n + " " + Arrays.toString(numbers));
        return numbers[1] * 2;
    }

    public static void mystery2(int x, int y) {
        x++;
        y = y * 3;
    }
}
```

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++) {
        if (list[i] <= list[i + 1]) {
            list[i + 1] = i * 2;
        }
    }
}
```

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
{4, 1, 3}	________________
{2, 1, 3, 2}	________________
{3, 6, 2, 9, 5}	________________
{1, 1, 1, 1, 8}	________________
{1, 3, 4, 6, 2, 9}	________________

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

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

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

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

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

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

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

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

Consider the following code fragment:

```
    C[] elements = {new B(), new C(), new A(), new D()};
    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 reportScore that takes as a parameter a Scanner containing information about a student's performance and that prints a report for that student. Students are given gold stars for tasks they solved particularly well and minuses for tasks where they performed poorly. The information for each student is presented as a series of count/type pairs where each count is a positive integer and each type is either "*" or "-". These count/type pairs are followed by the student's name which is guaranteed to be a single word composed entirely of alphabetic characters. Consider this report:

```
3 * 2 - 1 * Erica
```

It indicates that Erica got 3 stars followed by 2 minuses followed by 1 star. The overall score is computed by giving 1 plus point for each star and one minus point for each minus. The method produces exactly one line of output reporting this score and the total number of tasks. The table below includes several examples.

Scanner contents	Output produced
3 * 2 - 1 * Erica	Erica got 2 of 6
2 - 1 * 2 - 1 - 1 * 2 - Fred	Fred got -5 of 9
1 * 1 - 3 * 1 - 2 - Sylvia	Sylvia got 0 of 8
Julia	Julia got 0 of 0

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

5. Line-Based File Processing, 9 points. Write a static method called reverseAndFlip that takes a Scanner containing an input file as a parameter and that writes to System.out the same file with successive pairs of lines reversed in order and with the second line of each pair reversed. For example, if the input file contains the following:

```
Twas brillig and the slithy toves
did gyre and gimble in the wabe.
All mimsey were the borogroves,
and the mome raths outgrabe.

"Beware the Jabberwock, my son,
the jaws that bite, the claws that catch,
Beware the JubJub bird and shun
the frumious bandersnatch."
```

The method switches the order of the first two lines, printing the second line reversed. Then it switches the order of the third and fourth lines, print the fourth line reversed. And so on. It should produce this output:

```
.ebaw eht ni elbmig dna eryg did
Twas brillig and the slithy toves
.ebargtuo shtar emom eht dna
All mimsey were the borogroves,
,nos ym ,kcowrebbaJ eht eraweB"

nuhs dna drib buJbuJ eht eraweB
the jaws that bite, the claws that catch,
the frumious bandersnatch."
```

Notice that a line can be blank, as in the third pair. Also notice that an input file can have an odd number of lines, as in the one above, in which case the last line is printed in its original position. You may not make any assumptions about how many lines are in the Scanner and you may not construct any extra data structures to solve this problem.

6. Arrays, 10 points. Write a static method called isPairwiseSorted that takes an array of integers as a parameter and that returns whether or not the array is pairwise sorted. An array is considered to be pairwise sorted if it contains a sequence of pairs where each pair is in sorted (nondecreasing) order. For example, if a variable list is defined as follows:

```
int[] list = {3, 8, 2, 15, -3, 5, 2, 3, 4, 4};
```

then the call isPairwiseSorted(list) would return true because the array is composed of a sequence of pairs that are each in sorted order ((3, 8) followed by (2, 15), followed by (-3, 5), and so on). If the array has an odd length, then your method should ignore the value at the end. Below are several examples of what value would be returned for a given array.

Array passed as parameter	Value Returned
{}	true
{6}	true
{4, 12}	true
{8, 5}	false
{3, 8, 2, 15, -3, 5, 2, 3, 4, 4, 3, 1}	false
{8, 13, 92, 92, 4, 4}	true
{1, 3, 5, 7, 9, 8}	false

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

7. ArrayList, 10 points. Write a static method called reverse3 that takes an ArrayList of integer values as a parameter and that reverses each successive sequence of three values in the list. For example, suppose that a variable called list stores the following sequence of values:

```
[3, 8, 19, 42, 7, 26, 19, -8, 193, 204, 6, -4]
```

and we make the following call:

```
reverse3(list);
```

Afterwards the list should store the following sequence of values:

```
[19, 8, 3, 26, 7, 42, 193, -8, 19, -4, 6, 204]
```

The first sequence of three values (3, 8, 19) has been reversed to be (19, 8, 3). The second sequence of three values (42, 7, 26) has been reversed to be (26, 7, 42). And so on. If the list has extra values that are not part of a sequence of three, those values are unchanged. For example, if the list had instead stored:

```
[3, 8, 19, 42, 7, 26, 19, -8, 193, 204, 6, -4, 99, 2]
```

The result would have been:

```
[19, 8, 3, 26, 7, 42, 193, -8, 19, -4, 6, 204, 99, 2]
```

Notice that the values (99, 2) are unchanged in position because they were not part of a sequence of three values. 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 class called Ferret that extends the Critter class. The instances of the Ferret class always infect when an enemy is in front of them, otherwise hop if possible, and otherwise randomly choose between turning left and turning right (each choice equally likely). Their appearance changes based on whether they recently attempted to infect. They initially display as "I=0" indicating that they have not attempted to infect recently. After an infect move, they should display as "I=5". As the ferret makes other moves that are not infecting, this display should change to "I=4", "I=3", "I=2", "I=1", and "I=0". It should then stay at "I=0". Notice, however, that it can go back to "I=5" in the middle of this process because it might infect again before reaching "I=0". The ferrets should be blue when they are displaying as "I=0" and red otherwise.

   Use a Random object to make random choices but each Ferret should construct only one such object. As in assignment 8, fields must be declared private, fields initialized to a non-default value must be set in a constructor, and all updates to fields must occur in the getMove method.

9. Arrays, 15 points. Write a static method called splice that takes as parameters an array of integers and a "from index" (inclusive) and "to index" (exclusive) and that returns a new array containing the result of splicing together three segments of the array. The from and to indexes specify a sublist. For example, if list stores the following:

```
     [7, 5, 8, 5, 9, 7, 2, 3]
```

   then the call splice(list, 2, 4) indicates that the array should be rearranged using the sublist from 2 to 4:

```
         [7, 5]        [8, 5]    [9, 7, 2, 3]
     before sublist   sublist   after sublist
```

   The new array should contain the values after the sublist followed by the values in the sublist followed by the values before the sublist. For this example, it would return [9, 7, 2, 3, 8, 5, 7, 5].

   You may assume that the indexes passed as parameters specify a legal sublist of the list, although it might be empty. The method should not construct any extra data structures other than the array to be returned and it should not alter its array parameter. You are not allowed to call methods of the Arrays class to solve this problem.

10. Programming, 10 points. Write a static method called isMatch that takes a pattern string and a target string as parameters and that returns whether or not the given target matches the pattern. Patterns can contain special wildcard characters dot (".") and star ("*"). If a pattern does not contain any wildcards, then the target has to be the same string, as in isMatch("and", "and"). A dot can match any single character. For example, the pattern "a.." matches any 3-letter string beginning with the letter "a". A star can match any sequence of characters (including no characters). For example, the pattern "a*t" matches any string that begins with "a" and ends with "t", including "at". There will be at most one star in any given pattern, although a pattern can contain several dots and a star. Below are examples of patterns and matching strings (note that your method compares a pattern against a single string, not a list of strings).

Pattern	Matching strings
"hello"	hello
"..."	and, ant, but, cat, cow, hat, sat, tap, ten, the, tot
"a..."	atom, army, aunt, aura
".a.."	bats, task, yard, saga, lava
"...a"	tuna, soda, coma, aura, saga, lava
"....th"	growth, zenith, health
"a*"	a, an, at, and, ant, atom, aunt, apple, army, aura
"t*t"	tot, that, trot, tiniest
"the*"	the, then, there, therefore, thermal, thespians
".a*a"	saga, lava, saliva, tarantula, nausea
"t.e.p*"	twerp, trespass, thespians
".o*e."	poem, token, wolves, voucher, toothbrushes

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 listed on the cheat sheet (otherwise this problem can be solved in one line of code).

You can receive up to 6 points for a solution that handles the dot wildcard without handling the star wildcard. If you wish to pursue this option, please mark the box below:

```
+---+
|   | I agree to limit my score to 6 by handling just the dot wildcard
+---+
```

CSE142 Section Handout #9 Solutions

1. Reference Mystery. The program produces the following output.

```
        108 [5, 10, 14]
        8 [5, 10, 14]
        20 8
```

2.

```
        Original List                   Final List
        ---------------------------------------------------------------
        {4, 1, 3}                       {4, 1, 2}
        {2, 1, 3, 2}                    {2, 1, 2, 4}
        {3, 6, 2, 9, 5}                 {3, 6, 2, 4, 6}
        {1, 1, 1, 1, 8}                 {1, 1, 2, 1, 6}
        {1, 3, 4, 6, 2, 9}              {1, 3, 2, 4, 2, 8}
```

3. Inheritance. The output produced is as follows.

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

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

```
        public static void reportScore(Scanner input) {
            int score = 0;
            int count = 0;
            while (input.hasNextInt()) {
                int next = input.nextInt();
                String type = input.next();
                count += next;
                if (type.equals("*")) {
                    score = score + next;
                } else {
                    score = score - next;
                }
            }
            String name = input.next();
            System.out.println(name + " got " + score + " of " + count);
        }
```

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

```
        public static void reverseAndFlip(Scanner input) {
            while (input.hasNextLine()) {
                String first = input.nextLine();
                if (input.hasNextLine()) {
                    String second = input.nextLine();
                    for (int i = second.length() - 1; i >= 0; i--) {
                        System.out.print(second.charAt(i));
                    }
                    System.out.println();
                }
                System.out.println(first);
            }
        }
```

6. Arrays. One possible solution appears below.

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

7. ArrayLists. Two possible solutions appear below.

```
        public static void reverse3(ArrayList<Integer> list) {
            for (int i = 0; i < list.size() - 2; i += 3) {
                int n1 = list.get(i);
                int n3 = list.get(i + 2);
                list.set(i, n3);
                list.set(i + 2, n1);
            }
        }

        public static void reverse3(ArrayList<Integer> list) {
            for (int i = 0; i < list.size() - 2; i += 3) {
                list.add(i, list.remove(i + 2));
                list.add(i + 2, list.remove(i + 1));
            }
        }
```

8. Critters. One possible solution appears below.

```
        public class Ferret extends Critter {
            private int infectCount;
            private Random r;

            public Ferret() {
                r = new Random();
            }

            public Action getMove(CritterInfo info) {
                if (infectCount > 0) {
                    infectCount--;
                }
                if (info.getFront() == Neighbor.OTHER) {
                    infectCount = 5;
                    return Action.INFECT;
                } else if (info.getFront() == Neighbor.EMPTY) {
                    return Action.HOP;
                } else {
                    int choice = r.nextInt(2);
                    if (choice == 0) {
                        return Action.LEFT;
                    } else {
                        return Action.RIGHT;
                    }
                }
            }

            public Color getColor() {
                if (infectCount > 0) {
                    return Color.RED;
                } else {
                    return Color.BLUE;
                }
            }

            public String toString() {
                return "I=" + infectCount;
            }
        }
```

9. Arrays. One possible solution appears below.

```
        public static int[] splice(int[] list, int from, int to) {
            int[] result = new int[list.length];
            int index = 0;
            for (int i = to; i < list.length; i++) {
                result[index] = list[i];
                index++;
            }
            for (int i = from; i < to; i++) {
                result[index] = list[i];
                index++;
            }
            for (int i = 0; i < from; i++) {
                result[index] = list[i];
                index++;
            }
            return result;
        }
```

10. Programming. One possible solution appears below.

```
        public static boolean isMatch(String pattern, String text) {
            int j = 0;
            for (int i = 0; i < pattern.length(); i++) {
                char ch1 = pattern.charAt(i);
                if (ch1 == '*') {
                    int diff = text.length() - pattern.length() + 1;
                    if (diff < 0) {
                        return false;
                    } else {
                        j += diff;
                    }
                } else {
                    if (j >= text.length()) {
                        return false;
                    }
                    char ch2 = text.charAt(j);
                    j++;
                    if (ch1 != '.' && ch1 != ch2) {
                        return false;
                    }
                }
            }
            return j == text.length();
        }
```

below is a solution to the 6-point problem:

```
        public static boolean isMatch(String pattern, String text) {
            if (text.length() != pattern.length()) {
                return false;
            }
            for (int i = 0; i < pattern.length(); i++) {
                char ch1 = pattern.charAt(i);
                char ch2 = text.charAt(i);
                if (ch1 != '.' && ch1 != ch2) {
                    return false;
                }
            }
            return true;
        }
```