CSE 142, Spring 2009, Final Exam

Name: _____

Section: _____ **TA:** _____

Student ID #: _____

Rules:

- You have 110 minutes to complete this exam.
  You will receive a deduction if you keep working after the instructor calls for papers.
- This test is open-book/notes.
- You may not use any electronic device of any kind including calculators.
- Your code will be graded on proper behavior/output, not on style.
- You do not need to write any `import` statements in your code.
- Do not abbreviate any code that you write.
- Do not abbreviate any answer asking for output (show the complete output).
- If you enter the room, you must turn in an exam and will not be permitted to leave without doing so.
- You must show your Student ID to a TA or instructor for your submitted exam to be accepted.

Good luck!

Problem	Description	Earned	Max
1	Array Simulation		10
2	Inheritance		7
3	Parameters / References		8
4	Token-Based File Processing		8
5	Line-Based File Processing		10
6	Arrays		10
7	ArrayList		8
8	Critters		15
9	Arrays		14
10	Programming		10
TOTAL	**Total Points**		**100**

1. Array Simulation, 10 points

Consider the following method:

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

Indicate in the right-hand column what values would be stored in the array after the method `arrayMystery` executes if the array in the left-hand column is passed as its parameter.

Original Contents of Array	**Final Contents of Array**
`int[] a1 = {};` `arrayMystery(a1);`	_____
`int[] a2 = {7};` `arrayMystery(a2);`	_____
`int[] a3 = {3, 1};` `arrayMystery(a3);`	_____
`int[] a4 = {5, 6, 8, 100};` `arrayMystery(a4);`	_____
`int[] a5 = {0, 2, 4, 0, 10};` `arrayMystery(a5);`	_____

2. Inheritance, 7 points

Assume the following four classes have been defined:

```
public class Cephalopod extends Animal {
   public String toString() {
      return "a sea creature with " +
         tentacles() + " tentacles";
   }

   public void printMotion() {
      System.out.println("swims");
   }

   public String tentacles() {
      return "some";
   }
}

public class Octopus extends Cephalopod {
    public String tentacles() {
      return "8";
    }
}
```

```
public class Animal {
   public String toString() {
      return "some sort of animal";
   }

   public void printMotion () {
      System.out.println("unknown");
   }
}

public class Crow extends Animal {
   public String toString() {
      return "a crow";
   }

   public void printMotion() {
      System.out.println("flies");
   }
}

public class Squid extends Cephalopod {
   public String tentacles () {
      return "many";
   }
}
```

Given the classes above, what output is produced by the following code?

```
Animal[] a = {new Animal(), new Crow(), new Cephalopod(), new Octopus(), new Squid()};
for(int i = 0; i < a.length; i++) {
   System.out.println(a[i]);
   System.out.print("motion: ");
   a[i].printMotion();
   System.out.println();
}
```

3. Parameters and References, 8 points

The program below produces 4 lines of output. What are they?

```
import java.util.*;

public class Mystery {
  public static void main(String[] args) {
    int x = 1;
    int[] a = new int[4];

    x = x * 2;
    mystery(x,a);
    System.out.println(x + " " + Arrays.toString(a));

    x = x * 2;
    mystery(x,a);
    System.out.println(x + " " + Arrays.toString(a));
  }

  public static void mystery(int x, int[] a) {
    x = x * 2;
    if(x > 6) {
       a[2] = 14;
       a[1] = 9;
    } else {
       a[0] = 9;
       a[3] = 14;
    }
    System.out.println(x + " " + Arrays.toString(a));
  }
}
```

Line 1: _____

Line 2: _____

Line 3: _____

Line 4: _____

4. Token-Based File Processing, 8 points

Write a static method called `printStatistics` that takes as a parameter a `Scanner` containing a series of tokens. It should print out the sum of all the tokens that are legal integers, the sum of all the tokens that are legal doubles but not integers, and the total number of tokens of any kind. For example, if a `Scanner` called `data` contains the following tokens:

```
3 3.14 10 squid 10.x 6.0
```

and we make the following call:

```
printStatistics(data)
```

the method should print out:

```
int total: 13
double total: 9.14
total number of tokens: 6
```

For a `Scanner` with no tokens it should print:

```
int total: 0
double total: 0.0
total number of tokens: 0
```

5. Line-Based File Processing, 10 points

Write a static method called `netChange` that takes two parameters: a `Scanner` containing a series of lines that represent deposits and withdrawals to different bank accounts, as well as other banking events; and an `int` that is the account number to check. The method returns a `double` that is the "net change" to the balance for the account number given as a parameter. The lines relevant for computing the net change begin with the string *deposit* or *withdrawal*, followed by the account number matching the one to check, followed by the amount to deposit or withdraw. Deposits and withdrawals to other accounts should not affect your calculation. Also skip any lines that don't start with `"deposit"` or `"withdraw"`. For example, suppose you have an input file with the following lines:

```
deposit 1888 100.0
deposit 1234 100.0
Bank President indicted
withdraw 1888 50.0
deposit 1971 10000.0
asteroid hits head office
withdraw 1234 20.0
```

Then if the input above is stored in a `Scanner` called `input` and we evaluate the following statement:

```
double change = netChange(input, 1234);
```

the variable `change` should have the value 80.0, since we deposited $100.00 to account 1234 and then withdrew $20.00. We skip the activity for the other accounts and the problems with the asteroid and indictment.

You may assume each input line has at least one token and each input line beginning with "deposit" *or* "withdrawal" *is correctly formatted.*

6. Arrays, 10 points

Write a static method `averageWithNeighbors` that accepts one array `input` of real numbers and returns a new array such that each element of the result is the average of the same element of `input` and the neighboring elements of `input`. For example, if `input` refers to an array holding `{1.2, 2.7, 9.0, 0.3, 0.0}` then the result would be an array with `{1.95, 4.3, 4.0, 3.1, 0.15}` because 1.95 is the average of 1.2 and 2.7; and 4.3 is the average of 1.2, 2.7, and 9.0, and so on. Notice most elements have two neighbors, but two elements have one neighbor.

Your method may assume that `input` *has at least two elements.*

You should not modify the elements of `input`.

This table shows some additional calls to `averageWithNeighbors` and the expected values returned:

Input	Result Returned
`{2.0, 3.0, 4.0}`	`{2.5, 3.0, 3.5}`
`{0.0, 20.0}`	`{10.0, 10.0}`
`{5.0, 3.0, 1.0, -1.0, -3.0, -9.0}`	`{4.0, 3.0, 1.0, -1.0, -4.333, -6.0}`

7. Arrays, 14 points

Write a static method called `interleave` that takes two arrays `a` and `b` of `int`s as parameters, and returns a new `int` array. The length of `the result` should be the sum of the lengths of `a` and `b`. The elements in the result should be `a[0]`, then `b[0]`, then `a[1]`, then `b[1]`, and so forth, until you reach the end of the shorter array. If `a` is longer than `b`, then the remaining elements should be just the remaining elements in `a`; and vice versa if `b` is longer than `a`.

This table shows some calls to your method and the expected values returned:

Input		Result Returned
`{2, 3}`	`{10, 11}`	`{2, 10, 3, 11}`
`{}`	`{}`	`{}`
`{2, 3, 4, 5}`	`{10, 11}`	`{2, 10, 3, 11, 4, 5}`
`{2, 3}`	`{10, 11, 12, 13}`	`{2, 10, 3, 11, 12, 13}`

8. Programming, 10 points

Write a static method called `compress` that takes an `ArrayList` of `String`s as a parameter. It should replace each sequence of two or more equal `String`s in the `ArrayList` with a single `String` consisting of the number of `String`s that were equal, an asterisk, and the original `String`. For example, suppose that the parameter is an `ArrayList` named `a` containing:

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

Then after calling `compress` the `ArrayList a` should contain:

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

So the 3 occurrences of `"squid"` were replaced with `"3*squid"`, and the 2 occurrences of `"octopus"` were replaced with `"2*octopus"`. The two occurrences of `"clam"` aren't changed, since they aren't next to each other.