Points off	1	2	3A	3B	4A	4B	Total off	Net Score

CS 307 – Midterm 1 – Fall 2009

Your Name_____

Your UTEID _____

Circle yours TA's name: Oswaldo Rashid Swati

Instructions:

1. Please turn off or silence your cell phones.
2. There are **4** questions on this test.
3. You have 2 hours to complete the test.
4. You may not use a calculator or any other electronic devices while taking the test.
5. When code is required, write Java code.
6. When writing a method, assume the preconditions of the method are met.
7. When writing a method you may add helper methods if you wish.
8. When you complete the test show the proctor your UTID and give them the test and any scratch paper. Please leave the room quietly.

1. (2 points each, 30 points total) Short answer questions. Place your answers on the attached answer sheet. For code sample state the output. If the code would cause a syntax error answer "syntax error". If it would cause a runtime exception answer "exception". If it would result in an infinite loop answer "infinite loop".

A. What is output by the following code?

```
int[] list1 = {4, 2, -1};
int[] list2 = new int[3];
list1[1]++;
list2 = list1;
list1[1]++;
System.out.println(Arrays.toString(list2)); // prints out elements
```

B. What is output by the following code?

```
int[] list3 = {2, 1, 5, 1};
System.out.println( list3[ list3[0] ] );
```

For questions C – N consider the following classes and interfaces.

```
public interface TotalPoints{
      public int getPoints();
}

public class WinLossRecord implements TotalPoints{

      private int wins;
      private int losses;

      public WinLossRecord(){
            this(0, 0);
      }

      public WinLossRecord(int ws, int ls){
            wins = ws;
            losses = ls;
      }

      public void win(){    wins++;    }

      public void lose(){   losses++;  }

      public int getWins(){ return wins; }

      public int getLosses(){ return losses; }

      public int getPoints(){
            return wins * 2;
      }
}

public class WinLossRecordWithTies extends WinLossRecord{

      private int ties;

      public WinLossRecordWithTies(){
            ties = 0;
      }

      public WinLossRecordWithTies(int ws, int ls, int ts){
            super(ws, ls);
            ties = ts;
      }

      public void tie(){ ties++; }

      public int getTies(){ return ties; }

      public int getPoints(){
            return getWins() * 3 + getTies();
      }
}
```

C. State if the following declarations are valid or invalid (meaning they cause a syntax error). 1 point each

```
TotalPoints t1 = new TotalPoints();
Object obj1 = new WinLossRecord();
```

D. State if the following declarations are valid or invalid (meaning they cause a syntax error). 1 point each

```
WinLossRecordWithTies w2 = new Object();
TotalPoints t2 = new WinLossRecordWithTies(3, 2, 3);
```

E. State if the following declarations are valid or invalid (meaning they cause a syntax error). 1 point each

```
TotalPoints t3 = new WinLossRecord();
WinLossRecord w3 = new WinLossRecordWithTies(0, 4, 0);
```

F. What is output by the following code when method `f2` is called?

```
public void f1(WinLossRecord w){
     w.win();
     w.lose();
     w.win();
}

public void f2(){
     WinLossRecord ws = new WinLossRecord();
     f1(ws);
     System.out.println( ws.getPoints() );
}
```

G. What is output by the following code when method `g2` is called?

```
public void g1(int w){
     w += 3;
     w *= 2;
}

public void g2(){
     WinLossRecord ws = new WinLossRecord();
     g1( ws.getWins() );
     System.out.println( ws.getPoints() );
}
```

H. What is output by the following code when method `h2` is called?

```
public void h1(WinLossRecord w){
     System.out.print( w.getPoints() + " "); // single space added
}

public void h2(){
     WinLossRecord ws1 = new WinLossRecord(3, 5);
     h1(ws1);
     WinLossRecordWithTies ws2 = new  WinLossRecordWithTies(3, 5, 3);
     h1(ws2);
}
```

I. What is output by the following code?

```
WinLossRecordWithTies w5 = new WinLossRecordWithTies();
w5.win();
w5.win();
w5.tie();
System.out.println( w5.getPoints() );
```

J. What is the output by the following code?

```
TotalPoints[] tList = new TotalPoints[2];
tList[0] = new WinLossRecordWithTies();
tList[1] = new WinLossRecord();
System.out.println( tList[0].getPoints() + " " + tList[1].getPoints() );
```

K. Explain in one or two sentences why the following code contains a syntax error.

```
TotalPoints t4 = new WinLossRecord();
t4.win();
System.out.println( t4.getPoints() );
```

L. What is the output by the following code?

```
Object obj2 = new WinLossRecord();
WinLossRecordWithTies w6 = (WinLossRecordWithTies)obj2;
System.out.println( w6.getTies() );
```

M. Explain in one or two sentences why the following client code contains a syntax error.

```
WinLossRecord w7 = new WinLossRecord();
w7.win();
w7.lose();
w7.wins *= 2;
System.out.println( w7 );
```

N. What is output by the following client code?

```
WinLossRecord w8 = new WinLossRecord();
WinLossRecord w9 = new WinLossRecord();
System.out.println( w8 == w9 );
```

O. Recall the following methods from the `IntList` class we developed in lecture.

```
public class IntList{
     public IntList() // create an empty IntList

     public int add(int val) // add to end
     public int insert(int pos, int val) // add at specified location
     public int size() // return size of list

     public String toStiring();
     // returns String of the form [val1, val2, ..., lastVal]
}
```

What is output by the following client code?

```
IntList list = new IntList();
System.out.print( list.size() + " ");
list.add(5);
list.add(3);
list.add(0);
System.out.println( list.toString() );
```

2. The `IntList` class. (10 points) In lecture we developed an `IntList` class to represent a list of `int`s to demonstrate encapsulation and the syntax for building a class in Java. As discussed in class the internal storage container may have extra capacity.

Complete a method for the `IntList` class named `increaseRange`. This is an instance method in the `IntList` class that increases the values in a specified range of the calling object by the corresponding values in another `IntList`.

Here is the method header:

```
/*
pre: other != null
     0 <= start < size, start <= stop < size(),

post: all elements in this IntList in the range from start to stop
inclusive have been increased by the corresponding values in other.
The IntList other is not changed as a result of this call.
If an index in other does not exist in the specified range do not
alter the corresponding element in this IntList
*/
public void increaseRange(IntList other, int start, int stop){
```

Examples of calls to `increaseRange`:

Original `IntList`, resulting `IntList` on next line

```
[0, 3, 0, 2, 3, 4].increaseRange([2, 3, 2, 3], 1, 2)
     -> [0, 6, 2, 2, 3, 4]

[0, 3, 0, 2, 3, 4].increaseRange([2, 3, 2, 3], 0, 3)
     -> [2, 6, 2, 5, 3, 4]

[0, 3, 0, 2, 3, 4].increaseRange([2, 3, 2, 3], 0, 0)
     -> [2, 3, 0, 2, 3, 4]

[1, 2, 1, -2].increaseRange([-2, -3, -2, -4, -3], 0, 0)
     -> [-1, 2, 1, -2]

[1, 2, 1, -2].increaseRange([-2, -3, -2, -4, -3], 1, 3)
     -> [1, -1, -1, -6]

[3, -3, 3, 5, 3, 5].increaseRange([-5, 3], 0, 3)
     -> [-2, 0, 3, 5, 3, 5]

[3, -3, 3, 5, 3, 5].increaseRange([-5, 3], 1, 2)
     -> [3, 0, 3, 5, 3, 5]

[3, -3, 3, 5, 3, 5].increaseRange([-5, 3], 3, 5)
     -> [3, -3, 3, 5, 3, 5]
```

You may not use any other methods in the `IntList` class except for the `size` method unless you define and implement them yourself in this question. Recall that this method is in the `IntList` class and so you have access to all `IntList` objects' private instance variables.

You may not use objects or methods from other Java classes other than native arrays and the provided IntList methods and other IntList methods you define and implement yourself.

You are not required to check the preconditions of the method and when you write the method you may assume the preconditions of the method have been met.

```
public class IntList{

     private int[] container;
     private int listSize;

     // return the size of this list
     public int size()

/*
pre: other != null
     0 <= start < size, start <= stop < size(),

post: all elements in this IntList in the range from start to stop
inclusive have been increased by the corresponding values in other.
The IntList other is not changed as a result of this call.
If an index in other does not exist in the specified range do not
alter the corresponding element in this IntList
*/
public void increaseRange(IntList other, int start, int stop){
```

3. (30 points total) This question has two parts. It involves the `MathMatrix` class from assignment 3.

3A. (15 points) Complete a private instance method for the `MathMatrix` class that determines which columns in the 2d array of `int`s storing the coefficients of the system of linear equations the `MathMatrix` object represents contain all zeros. The method shall return an array of `boolean`s equal in length to the number of columns in the `MathMatrix` object. If a column contains all zeros the corresponding `boolean` value shall be set to true, otherwise it shall be set to false.

Recall the 2d array of `int`s named `coeffs` does not maintain any extra capacity and that it is a rectangular matrix.

Consider the following example.

0	0	4	0	10
0	2	0	0	5
0	3	5	0	2
0	5	5	0	-6

`coeffs` of the calling (`this`) `MathMatrix` object.

The method would return an array of `boolean`s of length 5 with elements 0 and 3 set to true and the other elements set to false: `{true, false, false, true, false}`

Complete the private instance method `findColumnsWithAllZeros` based on the following assumptions and constraints.

- Do not use any other Java classes when completing this method other than native arrays and the given methods of the `MathMatrix class.`
- Recall the number of rows in the `MathMatrix` object equals `coeffs.length` and the number of columns equals `coeffs[0].length`. There is no extra capacity.
- `MathMatrix` objects will have at least one row and at least one column per row.
- `coeffs` is always a rectangular 2d array.
- You will not get full credit if your code performs more checks than are necessary.
- The calling object is not changed as a result of the method.
- You do not need to check the preconditions or create code to deal with cases when the preconditions are not met.

```
public class MathMatrix{

     private int[][]coeffs;

     public int numRows(){ return coeffs.length; }

     public int numCols(){ return coeffs[0].length; }

     // complete the instance method on the next page.
```

```
/*
pre: coeffs != null, coeffs.length > 0, coeffs[0].length > 0
     coeffs is a rectangular 2d array.

post: return an array of booleans equal in length to the number of
columns in coeffs. Any columns that consist of all zeros have their
corresponding element in the array of booleans set to true. All other
elements in the array of booleans are set to false.
*/
private boolean[] findColumnsWithAllZeros(){
```

3B. (15 points) Complete an instance method in the `MathMatrix` that removes all columns from the calling `MathMatrix` object that consist entirely of zeros. Your method shall rely on the method `findColumnsWithAllZeros` that you wrote in part A of this question. Assume your `findColumnsWithAllZeros` method works regardless of what you wrote in part A.

Consider the following example.

0	0	4	0	10
0	2	0	0	5
0	3	5	0	2
0	5	5	0	-6

`coeffs` of the calling (`this`) `MathMatrix` object.

The resulting `MathMatrix` object's `coeffs` instance variable would be altered as shown:

0	4	10
2	0	5
3	5	2
5	5	-6

`coeffs` of the calling (`this`) `MathMatrix` object.

Complete the private instance method `removeColumsnWithAllZeros` based on the following assumptions and constraints.

- Do not use any other Java classes when completing this method other than native arrays and the given methods of the `MathMatrix` class.
- Recall the number of rows in the `MathMatrix` object equals `coeffs.length` and the number of columns equals `coeffs[0].length`. There is no extra capacity.
- `MathMatrix` objects will have at least one row and at least one column per row.
- `coeffs` is always a rectangular 2d array.
- You do not need to check the preconditions or create code to deal with cases when the preconditions are not met.
- You may assume there will be at least one column in this `MathMatrix` that is not all zeros.

```
// pre: none
// post: This MathMatrix object has had all columns that contains
// all zeros moved. Remaining columns are shifted to the left.
public void removeColumsnWithAllZeros(){
```

```
// More room on the next page:
```

```
// More room for removeColumsnWithAllZeros method
```

4 Working with objects (30 points total) This question has two parts. It involves the classes from Assignment 4, the NameSurfer assignment.

4A. (15 points) Complete a private instance method in the `Names` class that creates a frequency map of the letters that appear in the `ArrayList` of `NameRecord` objects, the single instance variable in the `Names` class.

Recall, a frequency map stores how many times a particular value appears. You are to count the number of times each of the letters 'a' through 'z' appears. If an uppercase letter appears convert it to a lower case letter. The method returns an array of `int`s with length 26. The first element stores the number of times 'a' and 'A' appear in the list of names, the second element stores the number of times 'b' and 'B' appear in the list of names, and so forth. Any non-letter characters in names are not counted. For example some names may contain digits or other non-letter characters. Those characters are to be ignored. Each letter in each name is counted once regardless of that name's ranks. For example the name "Emma" would add 1 to the 'a' count, 1 to the 'e' count, and 2 to the 'm' count regardless of the rank of "Emma" in any decade.

Here are the methods you may use in this question:

From the `NameRecord` class:

```
public String getName() // return the name for this record
```

From the `Character` class:

```
public static boolean isLetter(char ch)
//returns true if ch is an upper or lower case letter, false otherwise

public static boolean isUpperCase(char ch)
//returns true if ch is an upper case letter, false otherwise

public static char toLowerCase(char ch)
// returns the lowercase equivalent of ch if it exists, otherwise
// returns ch
```

From the `String` class:

```
public int length() // returns the length of this String

public char charAt(int pos) // returns the character at the specified
// position in this String. pre: 0 <= pos < length()
```

From the `ArrayList` class:

```
public int size() // return the number of elements in this ArrayList

public E get(int pos) // return the element at the specified position
// in this list. pre: 0 <= pos < size()
```

You can convert a lower case letter to an `int` between 0 and 25 with the following expression: (Assuming `ch` is a `char` between `'a'` and `'z'`.) `int index = ch - 'a';`

```
public class Names{

     private ArrayList<NameRecord> namesList;

     // pre: none. (namesList is never null)
     // post: return a frequency map of letters
     private int[] createFreqMapOfLetters() {
```

4B. (15 points) Complete a public instance method in the `Names` class that creates and returns an `ArrayList` of `String`s that are names from the database that contain both the most frequently used letter and the least frequently used letter (one or more times) based on the frequency map created in part A. For example if the most frequently used letter were 'e' and the least frequently used were 'x' and the names "Xena", "Xeena", "Max", and "Emma" were in the database of names, the result would contain "Xena" and "Xeena" but not "Max" or "Emma" . If there is a tie for most and / or least frequently used letter in the database of names use the one closest to 'a'.

Your method should rely on the method `createFreqMapOfLetters` that you wrote in part A of this question. Assume your `createFreqMapOfLetters` method works regardless of what you wrote in part A.

An `int` between 0 and 25 can be converted to the corresponding `char` between 'a' and 'z' with the following expression. (Assume `index` is an `int` variable between 0 and 25.):

```
char ch = (char)(index + 'a');
```

You may use the following methods on this question in addition to the methods listed in part A.

From the `ArrayList` class:

```
public ArrayList() // create a new empty ArrayList

public boolean add(E val) // add E to the end of this ArrayList
// always returns true
```

From the `String` class:

```
public int indexOf(char ch) // return the index of the first
// occurrence of ch in this String. If ch does not appear
// in this String returns -1.
```

From the `Character` class:

```
public static char toUpperCase(char ch)
// returns the uppercase equivalent of ch if it exists, otherwise
// returns ch


public class Names{

      private ArrayList<NameRecord> namesList;

      // pre: none. (namesList is never null)
      // post: return an ArrayList with the names in namesList that contain
      // the most and least frequently used characters one or more times.
      // If no names meet this criteria, an ArrayList of size 0 is returned.
      public ArrayList<String> namesWithMostAndLeastUsedLetter(){

// complete the method namesWithMostAndLeastUsedLetter on the next page
```

```
// complete the method namesWithMostAndLeastUsedLetter here
```

Question 1 answer Sheet

Name_____

A. _____

B. _____

1.

2.

C. _____

1.

2.

D. _____

1.

2.

E. _____

F. _____

G. _____

H. _____

I. _____

J. _____

K. _____

L. _____

M. _____

N. _____

O. _____