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

CS 314 – Midterm 1 – Spring 2014

Your Name_____

Your UTEID _____

Instructions:

1. There are **4** questions on this test. 82 points available. Scores will be scaled to 200 points.
2. You have 2 hours to complete the test.
3. Place you answers on this test. Not the scratch paper.
4. You may not use a calculator or any other electronic devices while taking the test.
5. When writing a method, assume the preconditions of the method are met. Do not write code to check the preconditions.
6. On coding questions you may add helper methods.
7. When answering coding questions, ensure you follow the restrictions of the question.
8. Test proctors will not answer any questions regarding the content of the exam. If you think a question is ambiguous or has an error, state your assumptions and answer based on those assumptions.
9. When you complete the test show the proctor your UTID, give them the test and all the scratch paper, used or not, and leave the room quietly.

1. (1 point each, 20 points total) Short answer. Place your answer on the line next to or under the question. Assume all necessary imports have been made.

   a. If a question contains a syntax error or other compile error, answer "Compile error".
   b. If a question would result in a runtime error or exception answer "Runtime error".
   c. If a question results in an infinite loop answer "Infinite loop".
   d. Recall when asked for Big O your answer should be the most restrictive correct Big O function. For example Selection Sort has an average case Big O of $O(N^2)$, but per the formal definition of Big O it is correct to say Selection Sort also has a Big O of $O(N^3)$ or $O(N^4)$. I want the most restrictive, correct Big O function. (Closest without going under.)

A. What is the T(N) of `methodA`? Recall, T(N) is the function that represents the *actual* number of executable statements for an algorithm. N = `data.length`

```
public int methodA(int[] data) {
      int r1 = 0;
      for(int i = 0; i < data.length; i++)
            r1 += data[i] + i;
      int r2 = 0;
      for(int i = 0; i < data.length; i++)
            r2 += data[i] * i;
      return r1 - r2;
}
```

_____

B. What is the order (Big O) of `methodB`? N = `data.length` _____

```
public int methodB(int[] data) {
     int t = 0;
     for(int i = 0; i < data.length; i++)
          for(int j = data.length - 1; j >= i; j--) {
               t += data[i] - data[j];
               t += data[i] - j;
          }
     return t;
}
```

C. What is the worst case order (Big O) of `methodC`? N = `data.length`

_____

```
// data != null
public ArrayList<String> methodC(String[] data) {
     ArrayList<String> result = new ArrayList<String>();
     for(int i = 0; i < data.length; i++) {
          String temp = data[i];
          if(temp.length() > 10) {
               result.add(0, temp) ;// ArrayList insert method
               result.add(0, temp);
          }
     }
     return result;
}
```

D. What is the order (Big O) of `methodD`? N = `data.length` _____

```
public int[] methodD(int[] data, int scale) {
     int[] result = new int[data.length / 3];
     int ir = 0;
     for(int i = 10; i < data.length; i += 3) {
          result[ir] = data[i];
          data[i] = data[i] * scale;
          ir++;
     }
     data[0] += ir;
     result[result.length - 1] += ir;
     return result;
}
```

E. A method is $O(N\log_2 N)$. It takes 20 seconds for the method to run when N = 1,000,000. What is the expected time in seconds for the method to run when N = 4,000,000. Simplify your answer.

_____

F. A method is $O(N^2)$. It takes 25 seconds for the method to run when N = 100,000. What is the expected time in seconds for the method to run when N = 200,000?

_____

G. A method is $O(2^N)$. It takes 10 seconds for the method to run when N = 50. What is the expected time in seconds for the method to run when N = 60?

_____

H. What is the worst case order (Big O) of `methodH`? N = `data.length` _____

```
// data != null, data.length > 10;
public int methodH(int[] data, int tgt) {
     final int LIMIT = data.length / 2 + 5;
     int r = 0;
     for(int i = data.length / 2 - 5; i < LIMIT; i++)
          for(int j = i + 1; j < LIMIT; j++)
               if(data[i] * data[j] == tgt)
                    r++;
     return r;
}
```

I. What is the order (Big O) of `methodI`? N = `data.length` _____

```
public int methodI(int[] data, int tgt) {
     int t = 0;
     for(int i = 0; i < data.length; i++)
          for(int j = 1; j < data.length; j *= 3) {
               int t1 = data[i];
               int t2 = data[j];
               t += (t1 - t2) + (t1 * t2) + (t2 + t1);
          }
     return t;
}
```

J. What is output by the following code? _____

```
ArrayList<Integer> list1 = new ArrayList<Integer>();
list1.add(12);
list1.add(5);
list1.add(12);
list1.add(17);
list1.remove(1);
System.out.println(list1);
```

Refer to the classes defined on the sheet at the back of the test. You may detach that sheet for easier reference.

K. What is output by the following code? _____

```
Active ak = new Active(20, 5);
ak.print();
System.out.print(" " + ak.getSize()) ;
```

L. For each line of code write **valid** if the line will compile without error or **invalid** if it causes a compile error. (.5 points each)

```
Button b1 = new Focusable(20, 10);
```
_____

```
Passive p1 = new Label(10, "help");
```
_____

M. For each line of code write **valid** if the line will compile without error or **invalid** if it causes a compile error. (.5 points each)

```
Object obj1 = new Passive(10);
```
_____

```
Button b2 = new Label(10, "ok");
```
_____

N. What is output by the following code? _____

```
Component cn = new Button(30, 5);
cn.print();
System.out.print(" " + cn.getSize());
```

O. What is output by the following code? _____

```
Focusable f = new Button(20, 10);
f.print();
System.out.print(" " + f. getFocusColor());
```

P. What is output by the following code? _____

```
Button b4 = new Button(20, 10);
b4.print();
System.out.print(" " + b4. getFocusColor() + " " + b4.getSize());
```

Q. What is output by the following code?_____

```
Passive p3 = new Label(30, "help");
p3.print();
```

R. In one sentence, explain why the following code will not compile:

_____

```
ArrayList<Component> ar3 = new ArrayList<Component>();
ar3.add(new Component(10));
ar3.add("Label");
ar3.add(new Passive(20));
```

S. In one sentence, explain why the following class will not compile:

_____

```
public class Switch extends Active implements Comparable<Switch> {

     public boolean on;

     public Switch(int s, int m) { super(m, s); }

     public void print() {
          super.print();
          System.out.print(on);
     }
}
```

T. What is output by the following code? _____

```
ArrayList<String> ar3 = new ArrayList<String>();
ar3.add("C");
ar3.add("D");
ar3.add("B");
ar3.add("A");

Iterator<String> it = ar3.iterator();
System.out.print(it.hasNext());
System.out.print(" " + it.next());
it.remove();
System.out.print(" " + it.next());
```

2. The `GenericList` class. (17 points) To demonstrate encapsulation and the syntax for building a class in Java, we developed a `GenericList` class that can store elements of any data type. Recall our `GenericList` class stores the elements of the list in the first N elements of a native array. An element's position in the list is the same as the element's position in the array. The array may be larger than the list it represents.

Complete an instance method for the `GenericList` class, `int lengthOfStartingMatch(GenericList<E> other)`, that returns the number of elements at the start of the two lists that match. In other words the method returns the length of the shared prefix of the lists.

Note, the lists may contain elements equal to `null`.

Examples of calls to `lengthOfStartingMatch`.

```
[].lengthOfStartingMatch ([]) -> returns 0
["B", "A"].lengthOfStartingMatch ([]) -> returns 0
[].lengthOfStartingMatch (["A", "B"]) -> returns 0
["A", "B", "A"].lengthOfStartingMatch (["A","A","A","A"]) -> returns 1
["A", "C", "B"].lengthOfStartingMatch (["A","C","B","A"]) -> returns 3
["A", "C", "B", "E"].lengthOfStartingMatch (["A", "C"]) -> returns 2
["A", null, "B", "E"].lengthOfStartingMatch (["A", null, "E", "B"])
                                                    -> returns 2
[null, "A", "A", "B", "A"].lengthOfStartingMatch(["A", "A", "A"])
                                                    -> returns 0
```

You may not use any other methods from the `GenericList` class unless you define and implement them yourself as part of your answer.
You may not use other Java classes or methods, except native arrays and the `equals` method.
Your method shall be as efficient as possible given the constraints.

The `GenericList` class:

```
public class GenericList<E> {

     private E[] values;
     private int size;
```

Complete the following instance method for the `GenericList` class.

```
/*   pre: other != null
     post: Returns the number of elements at the start of the lists
     that match exactly. Neither this GenericList nor other are
     altered as a result of this method call.
*/
public int lengthOfStartingMatch (GenericList<E> other) {

/*   pre: other != null
```

```
     post: Returns the number of elements at the start of the lists
     that match exactly. Neither this GenericList nor other are
     altered as a result of this method call.
*/
public int lengthOfStartingMatch(GenericList<E> other) {
```

3. `NameRecord` and `Names` class. (24 points total) Part A, (6 points): Write an instance method for the `NameRecord` class that returns `true` if any its ranks are greater than a given value.

The `NameRecord` class for this question:

```
public class NameRecord {
    private static final int UNRANKED = Integer.MAX_VALUE - 1000;

    private ArrayList<Integer> ranks;
    private String name;
```

The method you are writing returns `true` if any of the `NameRecord's` ranks are greater than a given value. Consider this example. For the given `NameRecord` the method returns `true` for a cutoff of 100. The method returns `false` for a cutoff of 400.

```
Amy 196 244 284 356 331 136 35 2 16 61 107
```

Complete the following instance method for the **`NameRecord`** class.
You may use any methods from the Java `ArrayList` class and the `Iterator` interface (explicitly or implicitly), but no other Java classes.
You may not use other methods from the `NameRecord` class unless you implement them yourself.
Your method shall be as efficient as possible given the constraints.

```
/* pre: none
   post: return true if any ranks in this NameRecord are greater than cutoff.
     This NameRecord is not altered as a result of this method.
*/
public boolean anyRanksGreater(int cutoff) {
```

3B. (18 points) Write an instance method for the `Names` class that removes all `NameRecord`s from the `Names` object that have a rank greater than a given cutoff. The `NameRecord`s that are removed are placed in an `ArrayList<NameRecord>` and returned by the method.

Do not re-implement the functionality of part A, `anyRanksGreater`. Call that method as necessary in Part B.

You may use any of the methods and constructors from the Java `ArrayList` class, the `Iterator` interface (explicitly or implicitly), and the `anyRanksGreater` method from part A.
Your method shall be as efficient as possible in terms of time given the constraints of the question.

The `Names` class for this question:

```
public class Names {
    private ArrayList<NameRecord> records;
    // Complete this method.
    /* pre: cutoff > 1
       post: remove all NameRecords that have one or more ranks greater
       than the given cutoff from this Names object.
       The method returns the NameRecords removed in an Arraylist */
    public ArrayList<NameRecord> remove(int cutoff) {
```

4. Math Matrix (21 Points) If most of the values in a matrix are the same value, usually 0, the matrix is said to be *sparse*. Many problems in computing, science, and engineering involve large, sparse matrices. If most of the values in the matrix are equal to 0 it may be possible to save time and space by **not** using a 2d array of `int`s to represent the matrix. Instead, only the values that are **not** equal to zero are stored. If this is done the positional data about each non-zero element must also be stored. For example, consider the following matrix.

2	0	4	0	0
0	12	0	0	0
0	0	0	-131	0
0	0	5	0	2

The same matrix could be represented as follows: (Each set of numbers represents the row, column, and value for a non-zero element in the matrix.)

`ArrayList` of `SMEntry` objects
`[(0,0,2), (0,2,4), (1,1,12), (2,3,-131), (3,2,5), (3,4,2)]`

Override the toString method for a `SparseMatrix` class.

```
public class SparseMatrix {
     private int numRows;
     private int numCols;
     private ArrayList<SMEntry> nonZeroValues;

     public String toString() // complete this method for question 4
}
```

A `SparseMatrix` object that models the matrix shown above would have `numRows` = 4, `numCols` = 5, and an `ArrayList` of `SMEntry` objects as shown above.

The `SMEntry` objects are stored in row major order in the `ArrayList`. In other words the elements in the `ArrayList` appear in the order you would get scanning the matrix from right to left, top to bottom.

The `SMEntry` class stores information for a single non-zero element:

```
public class SMEntry {
     public int getRow() // returns row of this element
     public int getCol() // returns column of this element
     public int getVal() // returns value of this element. Is never 0
}
```

The specification for the `toString` method is simpler than the `MathMatirx` assignment. Each element is followed by a single underscore (even the last one in a row) and each row is followed by a newline.

The following shows the result of `toString` for the sample matrix above when displayed.

```
2_0_4_0_0_
0_12_0_0_0_
0_0_0_-131_0_
0_0_5_0_2_
```

Complete the instance method `toString` below. **The only classes you may use are the `ArrayList`, `SMEntry`, and `StringBuilder` classes. (Recall the `append` and `toString` methods of `StringBuilder`.) You may not use any other methods from the `SparseMatrix` class unless you implement them yourself as part of your answer.**

```
public String toString() {
     StringBuilder sb = new StringBuilder();
```

For questions K - S, consider the following classes and interface:

```
public class Component {
      private int size;

      public Component(int s) { size = s; }

      public void print() { System.out.print("C" + size); }

      public int getSize() { return size; }
}

public class Passive extends Component {
      public Passive(int s) { super(s);     }

      public void print() { System.out.print("P" + getSize()); }
}

public class Label extends Passive {
      private String text;

      public Label(int s, String t) {
            super(s);
            text = t;
      }

      public void print() { System.out.print("L" + text); }
}

public class Active extends Component {
      private int margin;

      public Active(int s, int m) {
            super(s + m * 2);
            margin = m;
      }

      public int getSize() { return super.getSize() + margin * 2; };
}

public interface Focusable {
      public String getFocusColor();
}

public class Button extends Active implements Focusable {
      public Button(int s, int m) {super(s, m); }

      public String getFocusColor() { return "RED"; }

      public void print() { System.out.print("B" + getFocusColor()); }
}
```