Points off	1	2	3	4	5		Total off	Net Score

CS 314 – Midterm 1 – Spring 2013

Your Name_____

Your UTEID _____

Circle yours TA's name: Donghyuk Lixun Padmini Zihao

Instructions:

1. There are **5** questions on this test. The test is worth 70 points. Scores will be scaled to 175 for grade center.
2. You have 2 hours to complete the test.
3. You may not use a calculator or any other electronic devices while taking the test.
4. When writing a method, assume the preconditions of the method are met. Do not write code to check the preconditions.
5. When writing a method you may add helper methods if you wish.
6. When answering coding questions, ensure you follow the restrictions of the question.
7. Test proctors will not answer any questions.
8. When you complete the test show the proctor your UTID, give them the test and any scratch paper, and please leave the room quietly.

1. (1 point each, 20 points total) Short answer. Place you answers on a Question 1 answer sheet.
    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.)

Bonus Question. 1 point extra credit. A Java `int` is 4 bytes. How many bits are in 4 bytes?

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 total = 0;
     for(int i = 0; i < data.length; i++)
          for(int j = 0; j < data.length; j++) {
               int temp = i * j / 10;
               total += temp;
          }
     return total;
}
```

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

C. What is the order (Big O) of `methodC`? N = `list.size()`

```
public int methodC(ArrayList<Integer> list) {
     int sum = 0;
     for(int i = 1; i < list.size(); i *= 2)
          sum += list.get(i);
     return sum;
}
```

D. What is the worst case order (Big O) of `methodD`?
N = `data1.length = data2.length`

```
// pre: data1.length == data2.length
public int methodD(int[] data1, int[] data2) {
     int total = 0;
     for(int i = 0; i < data1.length; i++) {
          total += data1[i];
          total += data2[i];
     }

     for(int i = 0; i < data1.length; i++) {
          if(data1[i] % 10 == 0)
               total -= 10;

          for(int j = i; j >= 0; j--)
               if(data1[i] % data2[j] == 2)
                    total += i * j;
     }
     return total;
}
```

E. What is the order (Big O) of `methodE`? N = `data.length`. Assume all elements in `data` are between -99 and 99 inclusive.

```
// pre: all elements of data between -99 and 99 inclusive.
public String methodE(int[] data) {
     String result = "";
     for(int i = 0; i < data.length; i++)
          result += data[i] + ", " + " ";
     return result;
}
```

F. A method is $O(N^2)$. It takes 5 seconds for the method to run when N = 20,000.
What is the expected time for the method to run when N = 40,000?

G. A method takes 2 seconds to run when processing 2,000 pieces of data. It takes the same method 16 seconds to run when processing 4,000 pieces of data. What is the most likely order (Big O) of the method given this timing data?

H. What is the worst case order (Big O) of `methodH`? N = `table.length`. `table` is a square 2D array, `table.length == table[0].length`. Method `comp` is O(N) where N = `table.length`.

```
public double methodH(double[][] table, int tgt) {
     double result = 0.0;
     final int START = table.length / 2 - 2;
     final int STOP = START + 2;
     for(int r = START; r <= STOP; r++)
          for(int c = START; c <= STOP; c++)
               if(comp(table, r, c))
                    result += table[r][c];
     return result;
}
```

I. A method is $O(N\log_2 N)$. It takes .001 seconds for the method to run when N = 1,000. What is the expected time for the method to run when N = 2,000? Your answer must be simplified.

J. What is output by the following code?

```
ArrayList<String> list1 = new ArrayList<String>();
list1.add("B");
list1.add("C");
list1.add(1, "A"); // insert method for ArrayList
list1.add(1, "CC");
list1.add(0, list1.remove(2));
System.out.println(list1);
```

K. What is output by the following code?

```
ArrayList<String> list2 = new ArrayList<String>();
String[] data = {"A", "C", "D", "A", "G"};
for(String s : data)
     list2.add(s);
Iterator<String> it = list2.iterator();
System.out.print(it.next() + " " + it.next() + " " + it.next() + " ");
it.remove();
it.remove();
System.out.println(list2);
```

For questions L - R consider the following classes and interfaces:

```
public interface Reservable {
      public int getNumSeats();
}

public abstract class Room {
      private String name;

      public Room(String n) { name = n;      }

      public abstract boolean isWired();

      public String toString() { return "Room: " + name; }
}

public class Lab extends Room {
      private String systems;

      public Lab(String n, String s) {
            super(n);
            systems = s;
      }

      public String toString() { return "Lab: " + systems;}

      public boolean isWired() { return true; }
}

public class Office extends Room {
      public Office() { super("off"); }
      public boolean isWired() { return false; }
}

public class Classroom extends Room implements Reservable {
      private int seats;

      public Classroom(String n, int s) {
            super(n);
            seats = s;
      }

      public boolean isWired() { return false; }
      public int getNumSeats() { return seats; }
}

public class InstructionalLab extends Lab implements Reservable {
      public InstructionalLab() { super("teach", "dual"); }

      public int getNumSeats() { return 24; }
}
```

L. State if each of the following line is **valid** if it will compile without error or **invalid** if it causes a compile error. (.5 points each)

```
Object obj1 = new Room("Aud"); // L.1
Object obj2 = new InstructionalLab(); // L.2
```

M. State if each of the following line is **valid** if it will compile without error or **invalid** if it causes a compile error. (.5 points each)

```
Room r1 = new Classroom("4.813", 35); // M.1
InstructionalLab b1 = new Lab("3rd floor", "BeBox"); // M.2
```

N. State if each of the following line is **valid** if it will compile without error or **invalid** if it causes a compile error. (.5 points each)

```
Reservable r2 = new Office("Grond"); // N.1
Reservable r3 = new InstructionalLab(); // N.2
```

O. What is output by the following code?

```
Room r4 = new Lab("Dun", "Amiga");
System.out.print(r4);
```

P. What is output by the following code?

```
InstructionalLab b2 = new InstructionalLab();
System.out.print(b2.isWired() + " " + b2);
```

Q. What is output by the following code?

```
Office o1 = new Office();
System.out.print(o1.isWired() + " " + o1);
```

R. What is output by the following code?

```
Room r5 = new Classroom("6.516", 16);
System.out.print(r5.isWired() + " " + r5.getNumSeats());
```

S. What is output by the following code?

```
Map<String, Integer> map = new HashMap<String, Integer>();
map.put("BC", 2);
map.put("AD", 3);
map.put("BC", map.get("AD"));
System.out.print(map.get("A") + " " + map.get("BC"));
```

T. What is output by the following code?

```
ArrayList<String> list5 = new ArrayList<String>();
list5.add("A");
list5.add("B");
ArrayList<String> list6 = list5;
list6.add("A");
list6.add("B");
list6 = new ArrayList<String>();
System.out.print(list5.size() + " " + list6.size());
```

2. The `GenericList` class. (13 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, `sublist(int start)`, that creates and returns a new `GenericList` of all the elements from position `start` to the end of the list.

Examples of calls to `sublist`.

```
[].sublist(0) -> returns []
[A, B].sublist(0) -> returns [A, B]
[A, B].sublist(1) -> returns [B]
[A, B].sublist(2) -> returns []
```

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

The `GenericList` class:

```
public class GenericList<E> {

      private E[] values;
      private int size;

      public GenericList(int cap) {
            values = (E[]) (new Object[cap]);
            size = 0;
      }
```

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

```
/*   pre: 0 <= start < size
     post: return a new GenericList with all elements in this
     GenericList from position start to the end of the list. The
     calling GenericList is not altered as a result of this method
     call.
*/
public void sublist(int start) {
```

3. The `MathMatrix` class. (14 points total) Write an instance method `multiplyByVector` for the `MathMatrix` class from assignment 2 that multiplies the elements of a `MathMatrix` by the values in an `ArrayList<Integer>`.

Consider this example with the following 2 x 3 `MathMatrix` as the calling object:

$$\begin{bmatrix} 2 & 3 & 1 \\ -1 & 5 & -2 \end{bmatrix}$$

If the size of the `ArrayList` equals the number of rows in the `MathMatrix` object, align the list with each column and multiply by the corresponding value. For example if the list is [2, 3] the `MathMatrix` above becomes:

$$\begin{bmatrix} 4 & 6 & 2 \\ -3 & 15 & -6 \end{bmatrix}$$

If the size of the `ArrayList` equals the number of columns in the `MathMatrix` object, align the list with each row and multiply by the corresponding value. For example if the list is [2, 5, 4] the `MathMatrix` at the top of this page becomes:

$$\begin{bmatrix} 4 & 15 & 4 \\ -2 & 25 & -8 \end{bmatrix}$$

If the `MathMatrix` is square and the size of the `ArrayList` equals the number of rows of the `MathMatrix` object, align the list with each column (as in the first example) and multiply by the corresponding value. For example if we have this `MathMatrix`

$$\begin{bmatrix} 2 & 3 \\ -1 & 5 \end{bmatrix}$$

and a list of [2, 3], then the `MathMatrix` becomes:

$$\begin{bmatrix} 4 & 6 \\ -3 & 15 \end{bmatrix}$$

If the size of the `ArrayList` does not equal either the number of rows or the number of columns in the `MathMatrix` do not alter the `MathMatrix`. For example if the list is [1, 2, 3, 4] and the `MathMatrix` is the one at the top of the page, the `MathMatrix` is unchanged.

Recall the `MathMatrix` class:

```
public class MathMatrix {

    private int[][] coeffs; // no extra capacity
```

Complete the instance method for the `MathMatrix` class on the next page.

- **You may not use any other methods from the `MathMatrix` class unless you implement them yourself as part of your answer.**
- **You may not use any other methods or classes from the Java standard library except the `ArrayList` class.**
- **Your method shall be as efficient as possible given the constraints.**

```
/*   pre: vector != null
     post: as described above
*/
public boolean multiplyByVector (ArrayList<Integer> vector) {
```

4. Working with `ArrayLists`, `NameRecord` (12 points total) Write an instance method for the `NameRecord` class from assignment 3 that determines if a name is *trendy*. A name is trendy if it is initially unranked for one or more decades and its initial rank (after being unranked) is better than or equal to a given cutoff rank.

For example if the given cutoff rank is 250, here are some of the names that are trendy with a starting decade of 1900 and 11 ranks.

```
Bobby 0 0 135 26 49 77 79 98 134 249 370
Brittney 0 0 0 0 0 0 0 0 92 85 147
Gage 0 0 0 0 0 0 0 0 0 250 167
Woodrow 0 72 282 321 313 465 632 800 0 0 0
```

The following names are **not** trendy based on the criteria, because they **do not begin** with one or more unranked decades or do not have an initial rank better than or equal to the cutoff (assuming it is 250) after starting with one or more unranked decades:

```
Wilbur 124 96 121 180 253 409 491 776 0 0 0
Justina 651 676 641 773 0 0 0 0 100 300 400
Liza 0 0 251 100 50 0 0 0 200 205 350
Lisa 0 0 0 0 350 0 0 0 202 215 355
```

The `NameRecord` class for this question:

```
public class NameRecord {

    private static final int YEARS_PER_DECADE = 10;
    private static final int UNRANKED = Integer.MAX_VALUE - 1000;

    private final int BASE_YEAR;
    private ArrayList<Integer> ranks;
    private String myName;

     // NOTE: Instead of storing 0's for unranked decades
     // the ArrayList ranks stores the value UNRANKED to
     // indicate a name was not ranked in a given decade.
```

- **Note you may not use any other methods from the `NameRecord` class in this question unless you implement them yourself as part of your answer.**
- **You may not use any other methods or classes from the Java standard library except the `ArrayList` class.**
- **Your method shall be as efficient as possible given the constraints.**

Complete the instance method for the `NameRecord` class on the next page.

```
/*   pre: cutoff > 0
     post: return true if this NameRecord meets the criteria for a
          trendy name as described above, false otherwise
*/
public boolean isTrendy(int cutoff) {
```

5. New Data Structures (11 points) A *mutliset* or *bag* is a data structure that is unordered (unlike a list) and allows duplicates (unlike a traditional set).

For example the bag (C, A, B, B, A) is equal to the bag (B, A, A, B, C).

Write an instance method for a `Bag` class that removes a single occurrence of a given target value from the `Bag`. Examples:

`(C, A, B, B, A).remove(A) -> resulting Bag (C, A, B, B)`

Note, the `Bag` is unordered so the result could also be `(C, B, B, A)` or any other permutation of `(C, A, B, B)`. Likewise, in the following examples, the resulting `Bag` could be any permutation of the values shown in the result.

`(C, A, B, B, A).remove(D) -> resulting Bag (C, A, B, B, A)`
`(C, A, B, B, A).remove(C) -> resulting Bag (A, B, B, A)`
`().remove(C) -> resulting Bag ()`

For this question the internal implementation of the `Bag` is similar to an array based list. The `Bag` class stores the N elements of the bag in the first N elements of a native array of `Objects`. (Similar to an array based list.) The array may be larger than the `Bag` it represents. From the client's perspective the elements are unordered. The relative order of elements may change when altering the `Bag`.

- **You may not use any other methods in the `Bag` class unless you define and implement them yourself as part of your answer.**
- **You may not use objects or methods from other Java classes, other than native arrays and the `equals` method.**
- **Your method shall be as efficient as possible in terms of T(N), the actual number of operations required, given the constraints.**

```
public class Bag {

     private Object[] elements;
     private int numberOfElements;

     // complete the following instance method on the next page:

     /*   pre: tgt != null
          post: If tgt is present in this Bag, a single occurrence
                of tgt is removed, numberOfElements is decremented, and
                the method returns true.
                If tgt is not present, this Bag is unaltered and the
                method returns false.
     */
     public boolean removeSingleOccurrence(Object tgt) {
```

```
/*   pre: tgt != null
     post: If tgt is present in this Bag, a single occurrence
           of tgt is removed, numberOfElements is decremented, and
           the method returns true.
           If tgt is not present, this Bag is unaltered and the
           method returns false.
*/
public boolean removeSingleOccurrence(Object tgt) {
```

Question 1 Answer Sheet.

Name_____ UTEID _____

Bonus question: _____

A. _____

B. _____

C. _____

D. _____

E. _____

F. _____

G. _____

H. _____

I. _____

J. _____

K. _____

L. 1.
2.
_____

M. 1.
2.
_____

N. 1.
2.
_____

O. _____

P. _____

Q. _____

R. _____

S. _____

T. _____