Points off	1	2	3	4			Total off	Net Score

CS 314 – Exam 1 – Spring 2016

Your Name_____

Your UTEID _____

Instructions:

1. There are **4** questions on this test. 90 points available. Scores will be scaled to 180 points.
2. You have 2 hours to complete the test.
3. Place you final answers on this test. Not on scratch paper. Answer in pencil.
4. You may not use a calculator or any other electronic devices while taking the test.
5. When answering coding questions, ensure you follow the restrictions of the question.
6. Do not write code to check the preconditions unless the question requires it.
7. On coding questions you may implement your own helper methods.
8. On coding questions make your solutions as efficient as possible given the restrictions of the question.
9. 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.
10. 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, 25 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 a function that represents the *actual* number of executable statements for an algorithm. N = `data.length`

```
public int methodA(int[] data) {                    _____________________
      int r1 = 0;
      int r2 = 0;
      int r3 = 0;
      for (int i = 0; i < data.length; i++) {
            if (data[i] < 0) {
                  r1++;
            } else {
                  r2++;
            }
            r3 += data[i];
      }
      return r1 * r2 - r3;
}
```

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

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

_____

```
// data != null
public ArrayList<Integer> methodC(int[] data, int tgt) {
     ArrayList<Integer> result = new ArrayList<Integer>(); // O(1)
     for (int i = 0; i < data.length; i++) {
          if (data[i] >= tgt) {
               result.add(0, data[i]); // inserts value at position
          }
     }
     return result;
}
```

D. What is the worst case order (Big O) of `methodD`? N = `list.size()` _____
Method `process` is O(N) where N is the length of the list argument sent to the method.

```
public double methodD(ArrayList<Double> list, double t) {
     double result = 0;
     for (int i = 0; i < list.size(); i++) {
          for (int j = i + 1; j < list.size(); j++) {
               result += process(list, i, j);
          }
     }
     return result;
}
```

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

```
// pre: data.length >= 10, no elements in data == null
//        no elements in data are the empty String
public int methodE(String[] data) {
     int result = 0;
     final int LIMIT = data.length - 2;
     for (int i = 2; i < LIMIT; i++) {
          for (int j = i - 2; j < i + 2; j++) {
               if (data[i].charAt(0) == data[i].charAt(0)) {
                    result++;
               }
          }
     }
     return result;
}
```

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

```
// pre: data != null
public int methodF(int[] data) {
     int result = 0;
     for (int i = 0; i < data.length; i++)  {
          for (int j = 0; j < data.length; j++) {
               for (int k = 1; k < data.length; k *= 3) {
                    result += data[i] * data[k] + data[j];
               }
          }
     }
     return result;
}
```

G. What is the best case order (Big O) of `methodG`? N = `data.length` _____

```
// pre: data != null
public String methodG(int[] data, int tgt) {
     String r1 = "";
     String r2 = "";
     for (int i = 0; i < data.length; i++) {
          if (data[i] < tgt) {
               r1 += data[i] + " ";
          } else {
               r2 += data[i] + ", ";
          }
     }
     String result = r1;
     if (r2.length() > r1.length()) {
          result = r2;
     }
     return result;
}
```

H. A method is $O(N^2)$. It takes 5 seconds for the method to run when N = 10,000 .
What is the expected time in seconds for the method to run when N = 20,000?

_____

I. A method is $O(N^3)$. It takes 1 seconds for the method to run when N = 2,000. How much data can the method process given 27 seconds?

_____

J. A method is $O(N\log_2 N)$. It takes 10 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 = 8,000,000?

_____

K. What is output by the following code? _____

```
ArrayList<String> listK = new ArrayList<String>();
listK.add("C");
listK.add("X");
listK.add("K");
listK.add(1, listK.get(2)); // position, value
listK.add(0, "J");
System.out.println(listK);
```

L. What is output by the following code? _____

```
ArrayList<Object> listEll = new ArrayList<Object>();
listEll.add("BA");
listEll.add(12);
listEll.add(1.5);
listEll.add(new ArrayList<String>());
System.out.println(listEll);
```

M. What is output by the following code? _____

```
TreeMap<String, Integer> mapM = new TreeMap<String, Integer>();
mapM.put("G", 5); // key, value
mapM.put("X", 5);
mapM.put("A", 3);
mapM.put("G", 9);
mapM.put("M", -1);
System.out.println(mapM);
// The Map toString takes the form:
// {key1=value1, key2=value2, ..., keyN=valueN}
```

N. Consider the following method :

```
public static Map<String, Integer> count(Scanner sc) {
     Map<String, Integer> result = new TreeMap<String, Integer>();
     while (sc.hasNext()) {
          String t = sc.next();
          Integer f = result.get(t);
          if (f == null) {
               result.put(t, 1);
          } else {
               result.put(t, f + 1);
          }
     }
     return result;
}
```

If the line

```
Map<String, Integer> result = new TreeMap<String, Integer>();
```

is changed to

```
Map<String, Integer> result = new HashMap<String, Integer>();
```

Will the method run slower, faster, or about the same?

_____

O. What is output by the following code? _____

```
int[] org = {-5, 3, 4, 0};
ArrayList<Integer> listO = new ArrayList<Integer>();
for (int x : org)
     listO.add(x);
Iterator<Integer> it = listO.iterator();
System.out.print(it.next() + " ");
it.next();
it.next();
System.out.print(it.next() + " ");
System.out.print(listO);
```

P. What is output by the following code? _____

```
int[] org2 = {-5, 3, 4, 0};
ArrayList<Integer> listP = new ArrayList<Integer>();
for (int x : org2)
     listP.add(x);
Iterator<Integer> it2 = listP.iterator();
it2.next();
it2.next();
it2.remove();
it2.remove();
System.out.print(listP);
```

For questions Q through Y, refer to the classes defined on the sheet at the back of the test. You may detach that sheet for easier reference.

Q. 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)

```
AppStore aq1 = new Apple();
```
_____

```
Feature fq2 = new Smart(50);
```
_____

R. 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)

```
Phone pr1 = new Android();
```
_____

```
Apple ar2 = new Phone(400);
```
_____

S. What is output by the following code? _____

```
Feature fs = new Feature();
System.out.print(fs.sound() + " " + fs.getCost());
```

T. What is output by the following client code? _____

```
Phone pt = new Apple();
System.out.print(pt.sound() + " " + pt.numApps());
```

U. What is output by the following code? _____

```
Phone pu = new Android();
System.out.print(pu.sound());
```

V. What is output by the following code?_____

```
Landline vp = new Landline();
System.out.print(vp.toString());
```

W. What is output by the following code? _____

```
Feature fw = new Feature();
System.out.print(fw.getCost() + " ");
if (fw.getClass() == null) {
     System.out.print(1);
} else {
     System.out.print(2);
}
```

X. What is output by the following code? _____

```
Phone px = new Phone(10);
Android ax = (Android) px;
System.out.print(ax.numApps());
```

Y. What is output by the following code? _____

```
Phone[] pyAr = {new Phone(), new Smart(10), new Feature(),
          new Apple()};

for (Phone py : pyAr) {
     System.out.print(py.sound() + " ");
}
```

2. The `GenericList` class. (20 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.

The method creates and returns a new `GenericList` that contains all the elements in the calling `GenericList` (this) and other between `start` inclusive and `stop` exclusive. The elements in the specified range from the calling `GenericList` appear first in the resulting, list followed, by the elements of `other`.

```
public GenericList<E> getDualSublist(GenericList<E> other, int start, int stop)
```

Examples of calls to `getDualSublist`.

```
[A].getDualSublist([B], 0, 1) -> returns [A, B]
[B, A].getDualSublist([C, D, E], 1, 2) -> returns [A, D]
[A, B, C, A, D].getDualSublist([U, V, W, X, Y], 2, 4) -> returns [C, A, W, X]
[A, B, A].getDualSublist([D, E, F], 0, 3) -> returns [A, B, A, D, E, F]
[A, B].getDualSublist([A, C, B, A, B], 3, 3) -> returns []
```

The `GenericList` class:

```
public class GenericList<E> {

      private static final int DEFAULT_CAPACITY = 10;

      private E[] container;
      private int size;

      public GenericList() {
            this(DEFAULT_CAPACITY);
      }

      public GenericList(int initialCapacity) {
            container = getArray(initialCapacity);
            size = 0;
      }


      // returns an array of the given capacity
      private E[] getArray(int capacity)
```

You may only use the method and constructors from the `GenericList` class shown above.

You may not use any other methods from the `GenericList` class unless you implement them yourself as a part of your solution.

You may not use any other classes besides `GenericList` and native arrays.

Recall, this method is in the `GenericList` class so you have access to the private instance variables of any `GenericList` in scope.

```
/* pre: other != null, 0 <= start < size(), start <= stop <= this.size()
                                            start <= stop <= other.size()
   post: return a new GenericList with all the elements from this
         list from start inclusive to stop exclusive followed by all the
         elements in other from start inclusive to stop exclusive.
         The size of the returned list = 2 * (stop - start).
*/
public GenericList<E> getDualSublist(GenericList<E> other, int start, int stop) {
       // Do not write code to check preconditions
```

3. The MathMatrix class. (20 points) Write an instance method for the `MathMatrix` class from assignment two that determines if a `MathMatrix` could be a *strictly diagonal matrix*.

The *main diagonal* of a matrix runs from the upper left cell to the lower right cell.

For this question we define a *strict diagonal matrix* as a square matrix with all elements on the main diagonal not equal to 0 AND all elements not on the main diagonal equal to 0.

```
diagonal matrix:     3   0   0   0
                     0  -1   0   0
                     0   0   8   0
                     0   0   0  -5
```

There are some matrices that *could be* strictly diagonal if we were allowed to reorder the rows.

Consider the following example:

```
m2:     0   0   5   0
        2   0   0   0
        0   0   0  -2
        0   5   0   0
```

`m2` does not appear to be a strictly diagonal matrix, but if we were allowed to change the order of the rows we could have the following strictly diagonal matrix

```
m2 alt:      2   0   0   0
             0   5   0   0
             0   0   5   0
             0   0   0  -2
```

Consider the following two examples which **could not** be rearranged to strictly diagonal matrices.

```
m3:     0   0   0   1        m4:     0   4   0   0
        2   0   0   0                0   0   0   1
        0   0   0   0                2   0   0   0
        0   3   4   0                0   0   0   5
```

Write a method that determines if a `MathMatrix` could be a strictly diagonal matrix. Note, the method returns `true` for a `MathMatrix` that already is a strictly diagonal matrix.

You may not any other methods from the MathMatrix class unless you complete them yourself as part of your answer.

You may create and use a single Java native array with elements of a primitive type as part of your method. You may not use any other Java classes or methods.

```
public class MathMatrix {

        private int[][] coeffs;
        // coeffs.length is the number of rows in this MathMatrix
        // coeffs[0].length is the number of columns in this MathMatrix
```

```
// pre: this is a square matrix
// post: return if this is or could be a strictly diagonal matrix
public boolean possibleDiagonal () {
```

4. New Data Structures (25 points) A *bag* or *multiset* is a data structure that is unordered (unlike a list) and allows duplicates (unlike a traditional set).

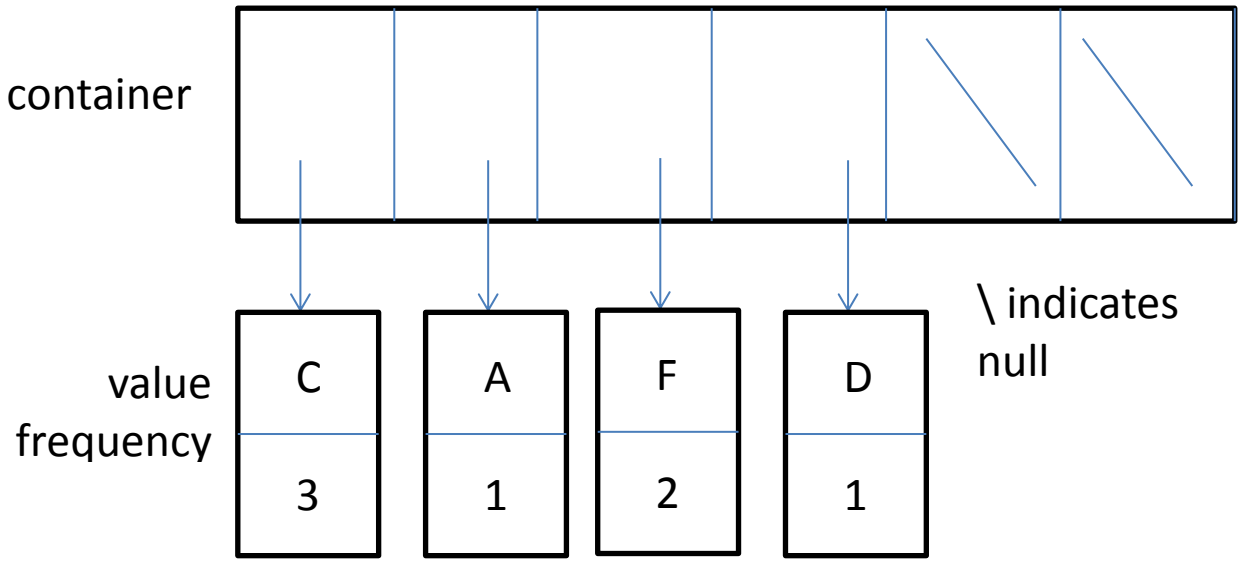
For example, the `Bag` **`(C, C, C, A, F, F)`** is equal to the `Bag` **`(C, C, F, A, F, C)`**

Write an instance method for a `Bag` class that adds a value to the `Bag`.

`(C, C, C, A, F, F).add(D) -> resulting Bag (C, C, C, A, F, F, D)`

In this implementation of the `Bag` class, the elements of the `Bag` are stored in a native array of `Pair` objects. Each `Pair` object stores a single `Object` and an int that represents how many times that `Object` is present in the `Bag`.

For example the `Bag` shown above could be stored as follows.



The `Bag` has one `Pair` object for each distinct element in the `Bag`. The `Pair` objects are in the first N elements of the array where N is the number of distinct items in the `Bag`.

There are instance variables for the total size of the `Bag` (total number of items) and the number of distinct items in the `Bag`. In the example above `sizeOfBag` = 6 and `distinctItemsInBag` = 3 before adding D. `sizeOfBag` = 7 and `distinctItemsInBag` = 4 after adding the D to 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 other Java classes or methods, other than native arrays, the given Pair class, and the `equals` method.**

Your method shall update all the instance variables of the `Bag` correctly.

Refer to the `Bag` and `Pair` classes on the next page and complete the `add` method for the `Bag` class.

```
public class Bag {

     private Pair[] container;
     private int sizeOfBag;
     private int distinctItemsInBag;

     /*   pre: value != null
          post: add the given value to this Bag
     */
     public void add(Object value) {
}

public class Pair {

     // create a new Pair with the given object and frequency
     public Pair(Object obj, int frequency)

     // get the Object from this Pair
     public Object getObject()

     // get the frequency of this Pair
     public int getFrequency()

     // set the frequency of this Pair
     public void setFrequency(int newFrequency)

}
```

Complete the add method for the Bag class on the next page.

```
/*   pre: value != null
     post: the given value has been added to this Bag
*/
public void add(Object value) {
```

For questions Q - Y, consider the following classes and interface:

```
public class Phone {
      private int cost;

      public Phone() {cost = 100;}

      public Phone(int c) {cost = c;}

      public int getCost() {return cost;}

      public String toString() {return "Phone: " + getCost();}

      public String sound() {return "default";}
}


public class Landline extends Phone {
      public String sound() {return "ring";}

      public int getCost() {return 25;}
}

public interface AppStore {
      public int numApps();
}

public class Smart extends Phone implements AppStore {
      public Smart(int cost) {super(cost);}

      public int numApps() {return 1000;}
}


public class Android extends Smart {
      public Android() {super(200);}

      public String sound() {return "tone";}
}


public class Apple extends Smart {
      public Apple() {super(500);}

      public String sound() {return "chime";}

      public int numApps() {return 500;}

      public int getCost() {return 500;}
}


public class Feature extends Phone {
      public String sound() {return "buzz";}
}
```