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

CS 314 – Exam 1 – Spring 2015

Your Name_____________________________________

Your UTEID __________________________________

Instructions:

1. There are **4** questions on this test. 75 points available. Scores will be scaled to 150 points.
2. You have 2 hours to complete the test.
3. Place you final 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 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, 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 t = 0;
     for(int i = 0; i < data.length; i++) {
          t += data[i] * 2;
          for(int j = 0; j < data.length; j++) {
               t+= data[i] - data[j];
          }
          t += i * data[i];
     }
     return t;
}
```

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 = 1; j < data.length; j *= 2)
               t += data[i] * data[j];
          for(int j = i + 1; j < data.length; j++)
               t += data[i] - data[j];
          t += data[i];
     }
     return t;
}
```

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

____

```
// data != null
public int methodC(int[] data, int c) {
     int t = 0;
     for(int i = 1; i < data.length; i++) {
          if(data[i] >= data[i - 1]) {
               for(int j = data.length - 1; j > 0; j = j / 2) {
                    if(data[j] < c)
                         t++;
               }
          }
     }
     return t;
}
```

D. What is the worst case order (Big O) of `methodD`? N = `list.size()` _________________

```
public double methodD(ArrayList<Double> list, double t) {
     double r = 0;
     for(int i = list.size() - 1; i >= 0; i--) {
          if(list.get(i) < t) {
               r += list.get(i);
               list.remove(i);
          }
     }
     return r;
}
```

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

____

F. A method is O(N). It takes 2 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 = 5,000,000?

____

G. Given the following timing data for an algorithm what is the most likely order (Big O) for the algorithm?

N	Time
1,000	2 second
2,000	16 seconds
4,000	128 seconds
8,000	1,024 seconds

____

H. What is output by the following code? ____

```
ArrayList<String> listH = new ArrayList<String>();
listH.add(0, "Q");
listH.add("P");
listH.add(0, "K");
listH.add(1, "P");
listH.add("X");
listH.remove(2);
System.out.print(listH);
```

I. What is output by the following code? ____

```
int[] dataI = {5, 8, -2, 0, 2, 4};
ArrayList<Integer> listI = new ArrayList<Integer>();
for(int xi : dataI)
     listI.add(xi);
Iterator<Integer> it = listI.iterator();
System.out.print(it.next() + " ");
it.remove();
System.out.print(it.next() + " ");
System.out.print(it.next() + " ");
it.remove();
System.out.print(listI);
```

J. What is output by the following code? _______________________________________

```
Map<Integer, String> mapJ = new TreeMap<Integer, String>();
mapJ.put(12, "A");
mapJ.put(3, "CC");
mapJ.put(7, "A");
mapJ.put(12, "C");
mapJ.put(5, "CC");
for(int k : mapJ.keySet())
     System.out.print(mapJ.get(k) + " ");
```

<u>For questions K through T, refer to the classes defined on the sheet at the back of the test. You may detach that sheet for easier reference.</u>

K. What is output by the following code? ______________________________

```
CrossListed ck = new CrossListed("CS", "M");
ck.add(5);
System.out.print(ck);
```

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)

```
Object obj1 = new AcademicClass("CS", 10);          _____________________

CrossListed c1 = new LabClass("BIO", 3);            _____________________
```

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)

```
AcademicClass a2 = new WritingClass("E", "CS", 2);     _____________________

CrossListed c3 = new AcademicClass("CH", 1000);        _____________________
```

N. What is output by the following code? ______________________________

```
AcademicClass an = new CrossListed("CS", "I");
an.add(10);
System.out.print(an);
```

O. What is output by the following client code? _______________________________

```
AcademicClass ao = new AcademicClass("GOV", 20);
methodO(ao);
System.out.print(ao);

public static void methodO(AcademicClass a) {
     a.add(10);
     a.add(a.getStudents());
}
```

P. What is output by the following code? _______________________________

```
AcademicClass ap = new IndependentStudy("HIS", "ECO", 5);
ap.add(5);
System.out.print(ap);
```

Q. What is output by the following code?_______________________________

```
CrossListed cq = new CrossListed("M", "CH");
System.out.print(cq instanceof AcademicClass);
```

R. What is output by the following code? _______________________________

```
AcademicClass acr = new AcademicClass("CS", 10);
LabClass cr1 = new LabClass("PHY", 3);
LabClass cr2 = new LabClass("BIO", 3);
System.out.print( acr.equals(cr1) + " " + cr1.equals(cr2));
```

S. What is output by the following code? _______________________________

```
AcademicClass as = new AcademicClass("HIS", 200);
as.add(100);
((CrossListed) as).add(100);
System.out.print(as);
```

T. What is output by the following code? _______________________________

```
LabClass ct = new LabClass("C", 3);
ct.add(25);
System.out.print(ct.equals(ct) + " " + ct);
```

2A. The `NameRecord` class (10 points)

Write an instance method for the `NameRecord` class from assignment 3 that returns an array of 2 ints.

The first element in the array is the best rank (closest to 1) for the given `NameRecord` and the second element is the worst rank (closest to unranked) for the given `NameRecord`.

If a name is unranked in a given decade this NameRecord class stores a 0 as the rank for that decade.

Consider the following examples with the data for a given `NameRecord` and the resulting array.

```
Tony 228 157 159 171 138 93 50 82 123 195 284        -> returns [50, 284]
David 29 30 22 11 6 5 2 4 5 11 16                     -> returns [2, 30]
Veronica 186 183 224 273 199 193 137 85 68 108 138    -> returns [68, 273]
Monte 0 0 922 463 395 334 360 557 852 0 0             -> returns [334, 1001]
```

Note, the method returns a 1001 as the worst rank if the `NameRecord` has any unranked decades.

The `NameRecord` class for this question:

```
public class NameRecord {

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

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

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

You may not use any other classes besides native arrays and the `ArrayList` class.

Your solution shall be efficient as possible given the constraints of the question.

```
/*
pre: none
post: returns an array of length 2.

The first element in the array is the best rank for this NameRecord
and the second element is the worst rank for this NameRecord.

If this NameRecord is unranked in any decade, returns 1001 as the
worst rank.
*/
public int[] getBestAndWorst() {
```

2B `Names` class (10 points)

We define `NameRecord` A to be ***constrained*** by `NameRecord` B if all of the ranks for `NameRecord` are between the best and worst ranks of `NameRecord` B.

In other words all the ranks of `NameRecord` A must be strictly greater than the best rank of `NameRecord` B **and** strictly less than the worst rank of `NameRecord` B.

Consider the following example. Assume `NameRecord` B is the `NameRecord` for Monte:

```
Monte 0 0 922 463 395 334 360 557 852 0 0
```

The best and worst ranks for this `NameRecord` are `[334, 1001]`.

The following `NameRecord`s **are** constrained by the `NameRecord` for Monte.

```
Ivy 362 512 761 707 952 585 758 633 614 526 352      -> [352, 952]
Lincoln 634 599 561 835 697 887 746 674 873 868 701  -> [561, 887]
Winston 580 484 425 484 376 461 582 641 676 672 708  -> [376, 708]
```

The following `NameRecord`s are **not** constrained by the `NameRecord` for Monte.

```
Wyatt 794 989 0 0 0 744 857 672 571 500 450        -> [450, 1001]
Olivia 315 311 290 307 270 355 504 324 193 47 16   -> [16, 504]
Troy 290 228 230 240 266 265 57 67 128 155 227     -> [57, 290]
```

The `Names` class for this question:

```
public class Names {

      private final int NUM_RANKS;

      // all elements of nameList have NUM_RANKS ranks
      private ArrayList<NameRecord> nameList;
```

The public interface of the `NameRecord` class for this question:

```
public class NameRecord {

      public int[] getBestAndWorst() // from part 2A

      // pre: 0 <= decade < numDecades()
      // post: return the rank for the given decade. Returns 0
            if unranked in the given decade
      public int getRank(int decade)
```

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

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

You may only use the methods from `NameRecord` shown on the previous page.

You may not use any other classes besides `NameRecord`, `ArrayList`, and native arrays.

Your solution shall be efficient as possible given the constraints of the question.

```
/*  pre: source != null
post: An ArrayList that contains all the NameRecords in this Names object
that are constrained by source. If there are none, the method returns an
empty ArrayList
*/
public ArrayList<NameRecord> constrained(NameRecord source)
       if(source == null) throw new IllegalArgumentException();
```

3. 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.

None of the elements in the `GenericList` will equal `null`.

```
public GenericList<E> inOtherListOnly(GenericList<E> other)
```

The method creates and returns a new `GenericList` that contains all the elements that are in parameter `other`, but not in the calling `GenericList` object.

Examples of calls to `inOtherListOnly`.

```
[].inOtherListOnly([]) -> returns []
["B", "A"]. inOtherListOnly([]) -> returns []
[].inOtherListOnly(["A", "B"]) -> returns ["A", "B"]
["A", "B", "A"]. inOtherListOnly(["B","A","B","A"]) -> returns []
["A", "B"]. inOtherListOnly(["A","C","B","A", "B"]) -> returns ["C"]
["A", "B", "B", "A"]. inOtherListOnly(["C", "B", "B", "A", "B", "C"])
      -> returns ["C", "C"]
```

The `GenericList` class:

```
public class GenericList<E> {

      private static final int DEFAULT_CAPACITY = 10;

      private E[] container;
      private int size;

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

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

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

You may only use the method and contructor 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.

You may use the `equals` method.

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

```
/* pre: other != null
   post: return a new GenericList with all the elements in other that
       are not in this GenericList. Neither this or other are changed.
*/
public GenericList<E> inOtherListOnly(GenericList<E> other) {
       if(other == null) throw new IllegalArgumentException();
```

4. Other data structures and iterators (15 Points)
A *Bag* or *Multiset* is an unordered collection of elements in which duplicates are allowed.

Because there is no notion of position for the client, the elements of a Bag may be stored in array with gaps present in the internal array.

For example the Bag (A, A, B, A, C) could be stored in the following array.
The lines represent references to the objects that are part of the Bag

```
[|, null, null, |, null, |, |, null, |, null]
 |              |        |  |        |
 A              A        B  A        C
```

Assume from a client's view, `null` elements are not allowed to be part of the Bag.
Therefore, a `null` in the array indicates an empty spot in the internal storage container.

To remove an element from the Bag simply set its corresponding reference to `null`.
For example, if we remove the first A in the Bag, the internal array becomes:

```
[null, null, null, |, null, |, |, null, |, null]
                   |        |  |        |
                   A        B  A        C
```

Implement a **complete** `Iterator` class for a `Bag` class.

```
public class Bag<E> implements Iterable<E> {

      private int size;
      private E[] container;

      public Iterator<E> iterator() { // required by Iterable interface
            return new BagIterator();
      }

      // inner class of Bag class
      private class BagIterator implements Iterator<E> {
            /* complete for this question*/  }

      // rest of class not shown
}
```

You may not use any other classes or methods in your solution except the required exception types and the methods you are implementing in the `BagIterator`.

Add instance variables as necessary.

Your solution shall be as efficient as possible given the constraints of the question.

Recall your BagIterator class must implement the following methods:

1. Returns `true` if there are more elements in the iteration
   `public boolean hasNext()`

2. Returns the next element in this iteration
   `public E next()`

3. Remove last element returned by this iteration from the underlying collection
   `public void remove()`

Recall the precondition to `next()` is `hasNext()`.
If the precondition is not met throw a `NoSuchElementException`.

Recall the `remove()` method throws an `IllegalStateException` if the `next()` method has not yet been called, or the `remove()` method has already been called after the last call to the `next()` method.

Complete the `BagIterator` with the necessary instance variables, zero argument constructor, `hasNext`, `next`, and `remove` methods. **On this question you must check preconditions.**

Complete this class on the next page.

```
// inner class of the Bag class
private class BagIterator implements Iterator<E> {
```

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

```
public class AcademicClass {
     private String department;
     private int students;

     public AcademicClass(String d, int s) {
          department = d;
          students = s;
     }

     public void add(int n) {    students += n; }

     public int getStudents() { return students; }

     public String toString() {  return department + getStudents(); }
}


public interface WritingClass {
     public int numPages();
}


public class CrossListed extends AcademicClass {

     private String otherDepartment;

     public CrossListed(String d1, String d2) {
          super(d1, 1);
          otherDepartment= d2;
     }

     public void add(int x) { super.add(x *2); }

     public String toString() {
           return otherDepartment + " " + getStudents();
     }
}


public class LabClass extends AcademicClass {

     private int labHours;

     public LabClass(String d, int h) {
          super(d, 0);
          labHours = h;
     }

     public boolean equals(LabClass c) {
          return this.labHours == c.labHours;
     }
} // another class on the next page
```

```
public class IndependentStudy extends CrossListed implements WritingClass {

      private int numPapers;

      public IndependentStudy (String d1, String d2, int n) {
            super(d1, d2);
            numPapers = n;
      }

      public int getStudents() { return 1; }

      public int numPages() { return numPapers * 10; }
}
```