Points off	1	2	3	4	5	6	Total off	Net Score

CS 314 – Exam 1 – Fall 2016

Your Name_____

Your UTEID _____

Instructions:

1. There are **6** questions on this test. 100 points available. Scores will be scaled to 180 points.
2. You have 2 hours to complete the test.
3. Place your 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, 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 a function that represents the *actual* number of executable statements for an algorithm. N = `data.length`

```
public double methodA(int[] data) {          _____________________
      int r = 0;
      final int LIMIT = data.length / 2;
      for (int i = 0; i < data.length; i++) {
            int s = data[i];
            for (int j = 0; j < data.length; j++) {
                  int t = data[j];
                  for (int k = 0; k < LIMIT; k++) {
                        r += s * t * data[k];
                  }
            }
      }
      return 1.0 * r / data.length;
}
```

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`

_____

```
// pre: data.length >= 5
public int methodC(int[] data, int tgt) {
     int r = 0;
     final int LIMIT = data.length - 3;
     for (int i = 2; i < LIMIT; i++)
          if (data[i] < tgt)
               for (int k = 1; k < data.length; k *= 3)
                    r += data[i] - data[k];
          else
               for (int k = i - 2; k <= i + 2; k++)
                    r += data[i] * data[k];
     return r;
}
```

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

Assume the `nextInt` method from the `Random` class is O(1).

```
public double methodD(int n, Random r) {
     double result = 0.0;
     for (int i = 0; i < n; i++) {
          double[] d = new double[n];
          int i1 = r.nextInt(n);
          int i2 = r.nextInt(n);
          int i3 = r.nextInt(n);
          result += d[i1] + d[i2] + d[i3];
     }
     return result;
}
```

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

```
public ArrayList<String> methodE(String[] data, int tgt) {
     ArrayList<String> r = new ArrayList<String>();
     for (int i = 0; i < data.length; i++) {
          if (data[i].length() < tgt)
               r.add(data[i]);
          else {
               r.add(data[i]);
               r.add(r.size() / 2, data[i]); // insert method
          }
     }
     return r;
}
```

F. What is the worst case order (Big O) of `methodF`? N = `list.size()` _____

```
public void methodF(ArrayList<Integer> list, int x) {
     Iterator<Integer> it = list.iterator();
     while (it.hasNext()) {
          if (it.next() < x) {
               it.remove();
          }
     }
}
```

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

```
// pre: data != null
public ArrayList<Double> methodG(ArrayList<Double> list, double tgt) {
     ArrayList<Double> result = new ArrayList<Double>();
     for (double d : list) {
          if (d > tgt) {
               result.add(0, d); // insert method
          }
     }
     return result;
}
```

H. A method is $O(N^2)$. It takes 2 second for the method to run when N = 2,000 .
What is the expected time in seconds for the method to run when N = 8,000?

_____

I. A method is $O(2^N)$. It takes 0.25 seconds for the method to run when N = 40.
What is the expected time in seconds for the method to run when N = 60?

_____

J. 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 = 2,000,000?

_____

K. Given the following timing data for a method what is the most likely order (Big O) of the method?

N (amount of data)	time to complete
1,000	1 second
2,000	16 seconds
4,000	256 seconds

_____

L. The following methods takes 5 seconds to complete when `list.size()` = 10,000.
What is the expected time for the method to complete when `list.size()` = 20,000

_____

```
public int methodL(ArrayList<Integer> list)
     int t = 0;
     for (int i = 0; i < list.size(); i++) {
          for (int j = list.size() - 1; j >= i; j--) {
               t += list.get(i) * list.get(j);
          }
     }
     return t;
}
```

M. What is output by the following code? _____

```
Map<Integer, Integer> mapM = new TreeMap<Integer, Integer>();
mapM.put(-5, 3); // key, value
mapM.put(3, 7);
mapM.put(0, -2);
mapM.put(3, 9);
mapM.put(7, mapM.get(3) + 3);
mapM.put(mapM.get(0) + 3, mapM.get(-5));
System.out.println(mapM);
// The Map toString takes the form:
// {key1=value1, key2=value2, ..., keyN=valueN}
```

N. What is output by the following code? _____

```
int[] data = {6, 3, 4, 7, 10};
ArrayList<Integer> list = new ArrayList<Integer>();
for (int x : data) {
     list.add(x);
}
Iterator<Integer> it = list.iterator();
while (it.hasNext()) {
     if (it.next() % 2 == 0) {
          System.out.print(it.next() + " ");
     }
}
```

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

O. 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 Student(12);
```
_____

```
Staff st1 = new Undergrad();
```
_____

P. 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)

```
UTPerson utp1 = new Faculty(3);
```
_____

```
GradStudent gs1 = new Object();
```
_____

Q. What is output by the following code? _____

```
Undergrad ug1 = new Undergrad();
System.out.print( ug1.toString() + " " + ug1.getPrimeTime());
```

R. What is output by the following code? _____

```
UTPerson utp2 = new Faculty(1);
System.out.print(utp2.toString() + " " + utp2.getHours());
```

S. What is output by the following code? _____

```
Student[] stus = new Student[2];
stus[0] = new GradStudent(10);
stus[1] = new Undergrad();
for (Student st : stus) {
     System.out.print(st + " ");
}
```

T. What is output by the following code? _____

```
UTPerson[] utps = {new Faculty(4), new Staff(), new GradStudent(6)};
for (UTPerson person : utps) {
     System.out.print( person.getPrimeTime() + " ");
}
```

2. The `GenericList` class (16 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.

Create an instance method for the `GenericList` class that returns the frequency of a given value in the list.

Examples of calls to the `frequency` method. (The values shown are `Integer` objects)

```
[].frequency(12) -> returns 0
[12, 1, null, 1, 13].frequency(37) -> returns 0
[12, 1, null, 1, 13].frequency(12) -> returns 1
[12, 1, null, 1, 13].frequency(1) -> returns 2
[12, 1, null, 1, 13].frequency(null) -> returns 1
```

Note, for this question, the `GenericList` may contain `null` values and the parameter itself may store `null`. You must deal with these possibilities correctly.

The `GenericList` class:

```
public class GenericList<E> {

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

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

You may call the `equals` method on objects.

Complete the method on the next page.

```
/// pre: none
// post: return the number of times val occurs in this list.
public int frequency(E val) {
```

3. GenericList (16 points) This question uses the `GenericList` class from question 2.

Create an instance method for the `GenericList` class `getNonMatchingPairs`. The method creates and returns a new `GenericList` that contains all the non matching pairs of items from the calling list and another list sent as a parameter.

Pairs of items are the elements of the two lists at the same index. For example given these two lists

```
       0  1  2  3
list1 [A, B, C, A]
list2 [A, A, C, D]
```

the pairs of items are (A, A), (B, A), (C, C), and (A, D). The non matching pairs are (B, A) and (A,D).

The method creates and returns a new list that contains the non matching pairs from the first two lists. The elements in the resulting list are the same order as the elements from the original lists with elements from the calling list coming before the element from the other list.

So given the two lists above, the method would return the following list: `[B, A, A, D]`

For this question you may assume **none** of the elements in either list equal `null`. If the lists are different sizes only paired elements are considered.

For example given these two lists

```
list1 [D, B, C, A, X, G]
list2 [A, B, Z, M]
```

the method shall return the following list: `[D, A, C, Z, A, M]`

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);
      }


      // 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.
You may call the `equals` method on objects.

```
// pre: other != null, no elements of this or other equal null
// post: per the problem description
public GenericList<E> getNonMatchingPairs(GenericList<E> other) {
```

4. Math Matrix (16 Points) Create a method for the `MathMatrix` class that returns `true` if the calling object is a *strictly diagonally dominant matrix*. A strictly diagonally dominant matrix is a square matrix where for every row in the matrix the value on the diagonal (from the upper left to the lower right) is greater than the sum of all the other elements in that row.

For example:

```
m1:  4    0    0      Not a strictly diagonally dominant because not a square matrix.
     0    3    -1
```

```
m2:  4    -3         A strictly diagonally dominant matrix.
     2    5
```

```
m3:  5    2    1    0      A strictly diagonally dominant matrix.
     2    13   -2   5
     0    0    1    0
     8    3    9    21
```

```
m4:  8    2    1    2      Not a strictly diagonally dominant matrix.
     9    13   -2   0
     2    3    9    5
     -2   -2   0    1
```

Recall the `MathMatrix` class:

```
public class MathMatrix {

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

Do not use any other Java classes besides `MathMatrix`.

You may not use any other methods from the `MathMatrix` class unless you implement them yourself as a part of your answer to this question.

Complete the method on the next page.

```
// pre: none
// post: return true if this MathMatrix is a strictly diagonally
// dominant matrix, false otherwise
public boolean isStrictlyDiagonallyDominant() {
```

5. Maps (16 points) Write a method named `indexMap` that accepts an array of `String`s as a parameter and returns a map with `String` keys and `ArrayList`s of `Integer`s values. The lists of `Integer`s are the indices in the original array at which the `String` occurred.

For example given an array with the following strings (quotes not shown):

```
[one, fish, two, fish, Sat, one, book]
```

the method shall return the following map:

```
{Sat=[4], book=[6], fish=[1, 3], one=[0, 5], two=[2]}
```

The map the method returns shall have the keys in sorted order and the list containing the indices shall contain the indices in sorted `String` order as in the example.

You may construct a Java `Map` (`TreeMap` or `HashMap`) and use the following `Map` methods:

put(key, value)	adds a mapping from the given key to the given value
get(key)	returns the value mapped to the given key (null if none)
containsKey(key)	returns true if the map contains a mapping for the given key
remove(key)	removes any existing mapping for the given key
clear()	removes all key/value pairs from the map
size()	returns the number of key/value pairs in the map

You may create and use `ArrayList`s.

Complete the method on the next page:

```
// pre: words != null, none of the elements of words == null
// post: pre the problem description
public static Map<String, ArrayList<Integer>> createIndexMap(String[] words)
{
```

6. Other Data Structures (16 points) In class implemented lists that store every value in the list explicitly. However, what if most of the elements of the list equal the same value? Consider the following list:

```
 0  1  2  3  4  5  6  7  8  9   10 11  12 13 14 15 16  // position
[A, B, A, A, A, A, A, A, A, A, AAA, A,  B, A, A, C, A] // element
```

Storing all those A's seems like a waste of space. In a ***sparse list***, only the elements not equal to the default value are explicitly stored. The default value is set when the list is created and does not change for a given list. Internally we use a native array as our storage container so we also store the position of each element, because the position in the array does not necessarily equal the position in the list.

Consider the following internal representation of the list shown above. Each element in the array is a `ListElem` object that stores one element of data and the position of that element in the list.
`(position, non-default element)`

The elements not equal to the default element, are stored in the array in ascending order based on their position in the list.

```
     0          1        2        3     index in array
[(1, B), (10, AAA), (12, B), (15, C)]   extra capacity not shown
size of list = 17
elements stored = 4
All elements not stored explicitly equal A for this list.
```

Complete the `E remove(int pos)` method for a `SparseList` class. The method removes the element at the given position from the list and updates instance variables appropriately.

Here is the `ListElem` class:

```
public class ListElem<E> {

     public ListElem(int position, E data) // create element

     public E getData() // return data of this element
     public int getPosition() // return position of this element

     public void setPositon(int pos) // set position of this element
     public void setData(E data) // set data of this element
```

The properties of the `SparseList` class are:

- the internal storage container is a native array of `ListElem` objects
- there may be extra capacity in the native array
- only elements not equal to the default element are stored explicitly
- the non-default elements are stored at the beginning of the array in ascending order based on their position in the list
- the size of the list, the number of elements stored explicitly in the array, and the default value are stored in separate instance variables

- any elements in the array that are not referring to active elements of the list are set to `null`
- the default list value never equals `null`

```
public class SparseList<E> {

     private ListElem<E>[] values;
     private int sizeOfList;

     // All values not stored explicitly in values equal defaultValue.
     // defaultValue never equals null.
     private final E defaultValue;

     // Number of elements stored explicitly in values.
     // The elements are stored at the beginning of the array.
     // This value could be 0 even if sizeOfList > 0 indicating
     // every element in the list is the default value.
     private int elementsStored;
```

Complete the `remove(int pos)` instance method for the `SparseList` class on the next page.

```
// pre: 0 <= pos < size()
// post: Remove and return element at given position in this list.
// All elements with position > pos are shifted one position to the left
// in this list. size() = old size() - 1
public E remove(int pos) {
```

For questions O - T, consider the following classes**. You may detach this sheet from the test.**

```
public abstract class UTPerson {
      private int hoursLimit;
      private int bookLimit;

      public UTPerson(int hours, int books) {
            hoursLimit = hours;
            bookLimit = books;
      }

      public abstract String getPrimeTime();

      public int getHours() {    return hoursLimit;    }

      public String toString() {
            return bookLimit + "-" + getHours();
      }
}

public class Staff extends UTPerson {

      public Staff() {    super(3, 5);      }

      public String getPrimeTime() {   return "8-5";    }
}

public abstract class Student extends UTPerson {

      public Student(int hours) {   super(hours, 10);  }

      public int getHours() {    return 15;       }
}

public class Undergrad extends Student {

      public Undergrad() {    super(20);    }

      public String getPrimeTime() {   return toString(); }
}

public class GradStudent extends Student {

      public GradStudent(int hours) {    super(hours); }

      public String getPrimeTime() {   return "12-2"; }

      public int getHours() {    return 10;       }
}
```

```
public class Faculty extends UTPerson {

	private int classes;

	public Faculty(int classes) {
		super(0, 30);
		this.classes = classes;
	}

	public String getPrimeTime() {	return "MWF";  }

	public int getHours() {	return classes * 5;	}

	public int officeHours() {	return classes * 3;	}

	public String toString() {
		return getPrimeTime() + officeHours();
	}
}
```