Points off	1	2	3	4A	4B	5	Total off	Net Score

CS 314 – Final Exam – Spring 2015

Your Name_____________________________________

Your UTEID _____________________________________

Instructions:

1. There are **5** questions on this test. 100 points available.
2. You have 3 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. When answering coding questions your solution must be as efficient as possible given the constraints 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 shall 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. A method uses the traditional insertion sort algorithm to sort an array of `int`s. It takes the method 2 seconds to complete when sorting an array of 20,000 distinct `int`s in random order. What is the expected time for the method to complete given an array of 80,000 distinct `int`s in random order?

____

B. A method is O(2^N). It takes the method 0.05 seconds to complete when N = 40. What is the expected time for method to complete when N = 60?

____

C. What is output when method `c()` is called?________________________

```
// The code uses the queue class we developed in lecture.
public void c() {
     int[] data = {42, 0, 13, 7, 20, 10, 5};
     Queue314<Integer> q = new Queue314<Integer>();
     for(int x : data)
          q.enqueue(x);
     cHelp(q);
     while(!q.isEmpty())
          System.out.print(q.dequeue() + " ");
}

public int cHelp(Queue314<Integer> q) {
     if(q.isEmpty())
          return 0;
     else {
          int t = q.dequeue();
          int s = cHelp(q);
          if(t < s)
               q.enqueue(t);
          return t + s;
     }
}
```

D. What is returned by the method call `d()` ? ________________________

```
public int d() {
     int[] tally = {0};
     int local = dHelp(tally, 4);
     return tally[0];
}

public int dHelp(int[] tally, int x) {
     tally[0]++;
     if(x <= 0)
          return 1;
     else
          return 3 + dHelp(tally, x - 1) + dHelp(tally, x - 2);
}
```

E. A graph is *sparse* if the number of edges is close to the minimum possible number of edges. A graph is *dense* if the number of edges is close to the maximum possible number of edges.

Are most real world graphs sparse or dense? ________________________

For F and G consider the following two classes:

```
public class MovieHouse {

     public void show() {     System.out.print(" M " + time());  }

     public int time() { return 150; }

     public int seats() { return 250; }
}

public class DraftHouse extends MovieHouse {

     public int time() { return super.time() + 30; }
}
```

F. What is output by the following code? _________________________

```
DraftHouse df = new DraftHouse();
System.out.print(df.time() + " " + df.seats());
```

G. What is output by the following code? _________________________

```
MovieHouse[] ms = {new DraftHouse(), new MovieHouse()};
for(MovieHouse m : ms)
     m.show();
```

H. What is output when method `h()` is called? _________________________

```
public void h() {
     int[] data = {5, 10};
     hHelp(data);
     System.out.print(Arrays.toString(data));
}

public void hHelp(int[] data) {
     int[] temp = data;
     for(int i = 0; i < data.length; i++)
          data[i] *= 2;
     System.out.print(Arrays.toString(data) + " ");
     data = temp;
}
```

I. The following values are inserted one at a time in the order shown to an initially empty, binary search tree using the naïve insertion algorithm.
What is the result of a pre-order traversal of the resulting tree?

`7, 2, -3, 5, 9, 6, -3`

____

J. What is the best case order (Big O) of the following method if the parameter `list` is a Java `ArrayList`? N = `list.size()`

____

```
public int j(List<Integer> list, int tgt) {
     int total = 0;
     for(int i = 0; i < list.size(); i++) {
          total += list.get(i);
          if(list.get(i) < tgt) {
               int temp = list.remove(i);
               list.add(0, temp); // (index, element)
          }
     }
     return total;
}
```

K. What is the best case order (Big O) of the method `j()` from 1.J if the parameter is a Java `LinkedList`? N = `list.size()`

____

L. What is the worst case order (Big O) of the method `j()` from `1.J` if the parameter is a Java `ArrayList`? N = `list.size()`

____

M. What is output by the following code?

```
int[] data = {6, 2, -3, 5, 2, 2, 6};
TreeSet<Integer> ts = new TreeSet<Integer>();
for(int x : data)
     ts.add(x);
for(int x: ts)
     System.out.print(x + " ");
```

N. The following method takes 5 seconds to complete when the `BinarySearchTree` class uses the traditional, naïve insertion algorithm and `n` = 10,000.
What is the expected time for the method to complete when `n` = 30,000?

____

```
public BinarySearchTree<Integer> tb(int n) {
     BinarySearchTree<Integer> r = new BinarySearchTree<Integer>();
     for(int i = 0; i < n; i++)
          r.add(i);
     return r;
}
```

O. Method `tb` from 1.N takes 10 seconds to complete when the `BinarySearchTree` class uses a Red-Black tree as its internal storage container and n = 1,000,000.
What is the expected time for the method to complete when `n` = 2,000,000?

____

P. The following values are inserted one at a time in the order shown to an initially empty Red-Back tree using the algorithm shown in class. Draw the resulting tree. Label each node as red or black.

`6, 4, 2`

Q. The following values are inserted one at a time in the order shown to a max heap using the algorithm shown in class modified for a max heap. Draw the resulting max heap.

```
8, 6, 4, 2, 6, 10
```

R. Consider the following method.

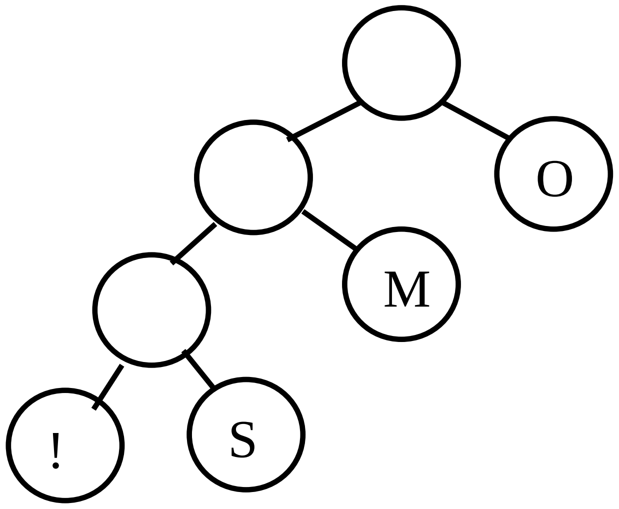
```
public Set<Double> rs() {
     final int LIMIT = 3000000;
     Random r = new Random(1970620);
     Set<Double> result = new TreeSet<Double>(); // line 1
     for(int i = 0; i < LIMIT; i++)
          result.add(r.nextDouble());
     return result;
}
```

If **`// line 1`** is changed to the following

```
Set<Double> result = new HashSet<Double>(); // line 1
```

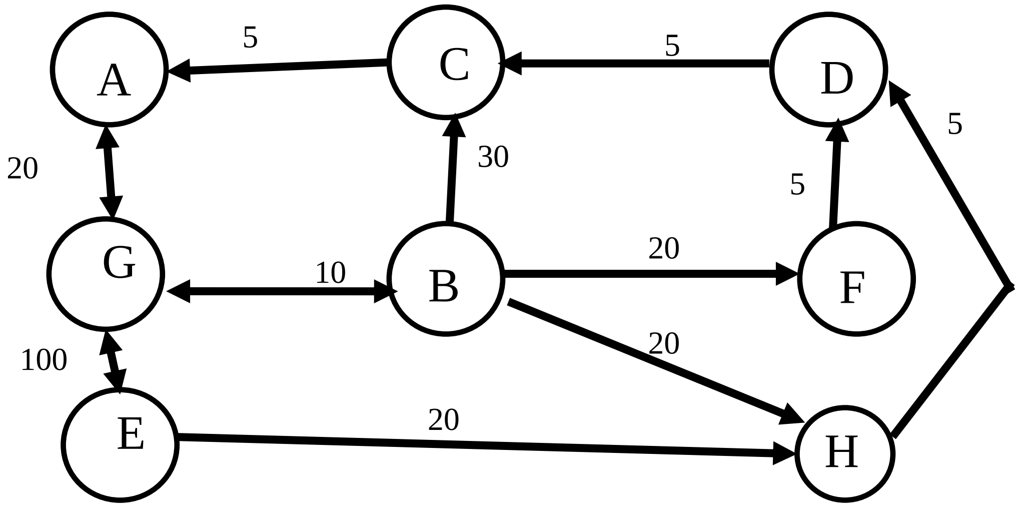
will the method take more time to complete, less time to complete, or about the same. Explain your choice in one sentence.

S. Consider the following Huffman code tree.



What is the encoding for **`MOM!`** given the above Huffman code tree?
Include a space between bits for clarity

T. Consider the following weighted, directed Graph:



What is the cost of the lowest cost path **from** vertex A **to** vertex E? ___________________

2. linked lists - 20 points. Complete the `removeRange` instance method for the `LinkedList314` class. The method removes all values between a start index inclusive and a stop index exclusive.

- You may not use any other methods in the `LinkedList314` class unless you implement them yourself as a part of your solution.
- The `LinkedList314` class uses singly linked nodes.
- The list only has references to the first node in the chained structure of nodes.
- When the list is empty, `first` is set to `null`.
- If the list is not empty, the last node in the list has its next reference set to `null`.
- You may use the given `Node` class. **You may not use any other Java classes or native arrays.**

```
public class LinkedList314<E> {
     private Node<E> first; // refers to first node
     private int size;      // number of elements and nodes in list
}
```

The `Node` class.

```
public class Node<E> {
     public Node(E item, Node<E> next)
     public E getData()
     public Node<E> getNext(
     public void setData(E item)
     public void setNext(Node<E> next)
}
```

Examples. of `removeRange(int start, int stop)`

```
[].removeRange(0, 0) ->  resulting list []

[A].removeRange(0, 0)  ->  resulting list [A]

[A].removeRange(0, 1)  ->  resulting list []

[A].removeRange(1, 1)  ->  resulting list [A]

[B, A, D, C, G, X].removeRange(2, 4)  ->  [B, A, G, X]

[B, A, D, C, G, X].removeRange(0, 5)  ->  [X]

[B, A, D, C, G, X].removeRange(1, 5)  ->  [B, X]

[B, A, D, C, G, X].removeRange(0, 6)  ->  []
```

Complete the following method instance method for the `LinkedList314` class.

```
/* pre:  0 <= start <= size(), start <= stop <= size
   post: remove the elements from start inclusive to stop exclusive
          from this list.
*/
public void removeRange(int start, int stop) {
```

3. binary trees - 20 points. Consider a binary tree that contains integers. The binary tree is ***not*** a binary search tree. Write a method that returns true if the root of every subtree is present only once (at the root itself) in its subtree. In other words, return true as long as the root of each subtree does **not** appear farther down in the subtree for which it is the root.

You may use the given `BinaryNode` class, but no other Java classes or methods. Not even arrays.

Consider the following trees and the expected return value.

```
empty tree -> returns true

5 -> returns true

   5 -> returns true         5 -> returns false
  / \                       / \   (duplicate 5)
 6   4                     5   4


        5       -> returns true
      /   \      duplicates exist in tree, but the roots of
     8    13     of subtrees DO NOT appear lower in their subtrees.
    / \     \
   13  6     8


        5       -> returns false
      /   \     8 (left child of 5) has duplicate 8 in its subtree
     8    13
    / \     \
   7   6    10
      / \
     4   9
    /
   8
```

Use the following `BinaryNode` class:

```
public class BinaryNode {
     public int getData();
     public BinaryNode getLeft();
     public BinaryNode getRight();
}
```

The given `BinaryTree` class:

```
public class BinaryTree {
     private BinaryNode root; // root == null if tree is empty
}
```

Hint: Add two helper methods.

Given the restrictions you method will likely be O(N^2)

Complete the `rootsNotInSubtrees` instance method for the `BianryTree` class.

```
/* pre: none
   post: per the question description. */
public boolean rootsNotInSubtrees() {
```

4. graphs - 20 points. This question has two parts.
Write two methods to determine the number of *subgraphs* in a graph.

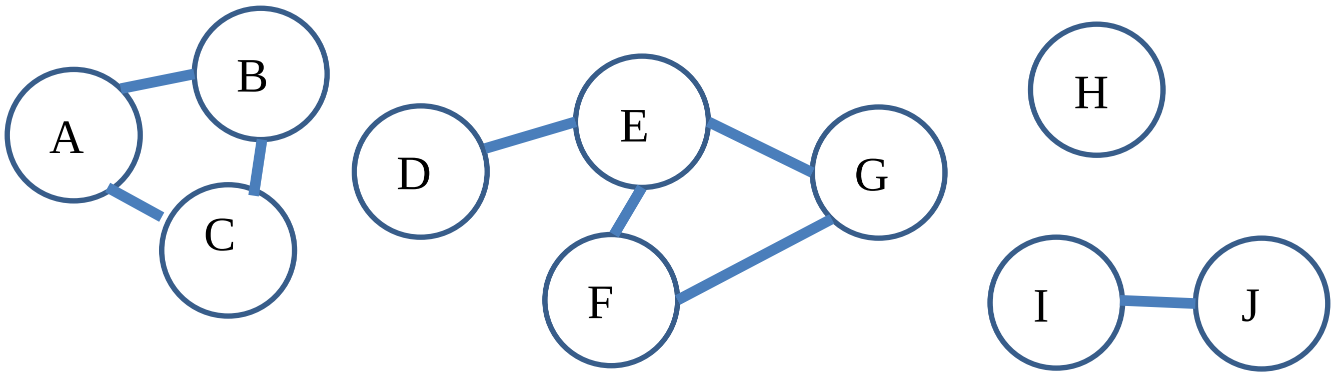
A graph is broken up into two or more subgraphs if the graph is not *connected*. Two vertices, A and B, are connected if there is a path in the graph from A to B. In the example below D and G are **connected** because there is a path between them even though no edge exists between D and G.

The graph itself is connected if every pair of vertices in the graph is connected.

If the graph is not connected it is broken up into subgraphs. A subgraph consists of 2 or more connected vertices **or** an isolated vertex (a vertex with zero edges).

Consider the graph below. It is a **single graph** with 10 vertices, A through J.

The graph has 4 subgraphs: [A, B, C], [D, E, F, G], [H], and [I, J]



Use the `Graph` class from assignment 12 and complete a method that returns the number of subgraphs in the calling `Graph` object.

In order to represent an undirected graph, if `Vertex` A contains an edge in its adjacency list to `Vertex` B, then `Vertex` B will have an edge in its adjacency list back to `Vertex` A.

Part A. 10 points.
Complete a private, recursive instance method that adds all vertices in a subgraph to a `Set` of `Vertex` objects.

On Part A you may use the `Vertex, Edge, List, Iterator`, and `Set` classes.
Do not create any new data structures on part A.

```
public class Graph {
    // The vertices in the graph.
    private Map<String, Vertex> vertices;

    private static class Vertex {
        private String name;
        private List<Edge> adjacent;
        private int scratch;
    }

    private static class Edge {
        private Vertex dest;
        private double cost;
```

```
/* pre: verts != null, current != null
   post: if verts does not contain current, add it and continue
     adding vertices that are part of the current subgraph.*/

private void addConnectedVertices(Set<Vertex> verts, Vertex current) {
```

Part B. 10 points
Complete an instance method in the `Graph` class that returns the number of subgraphs in the `Graph`.

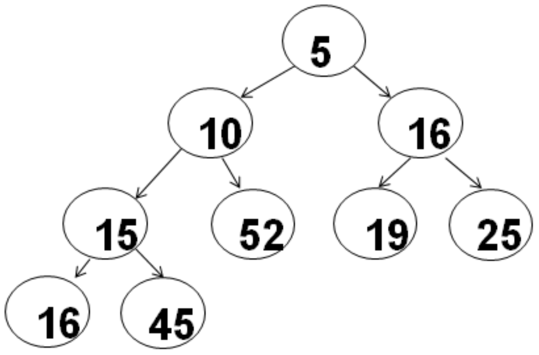
You may create and use a single `HashSet` object.

You may use the `Vertex`, `Edge`, `Iterator`, `HashSet`, and `Map` classes.

Call the `addConnectedVertices` method from part A. Do not re-implement that functionality.

```
//pre: none. post: return the number of subgraphs in this Graph.
public int getNumSubGraphs() {
```

5. data structures - 20 points. Write a method to decrease the value of a given element within a min heap that stores `int`s. Recall a min heap is a complete binary tree where each value is less than or equal to all descendant values. As demonstrated in class this min heap uses a native array to store values.



for element at position i:

parent index: *i* / 2

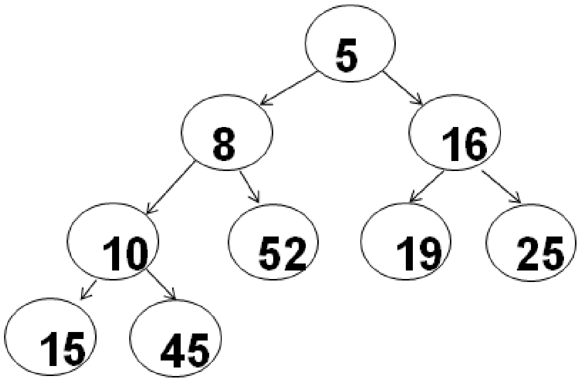
left child index: *i* * 2

right child index: *i* * 2 + *1*

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	5	**10**	**16**	**15**	**52**	**19**	**25**	**16**	**45**						

Implement this method:
```
/* pre: amount >= 0
   post: if element is present in this heap, it is decreased by amount
     and positioned correctly within the heap. If there are multiple
     copies of element in this heap then the deepest and rightmost
     instance of element.
*/
public void decreaseElement(int element, int amount)
```

The method finds the given element and reduces it by the given amount. The method then correctly repositions the altered element in the heap. After repositioning the altered element, the heap must be a complete tree with the min heap property restored. So for example, given the heap above, the method call `decreaseElement(16, 8)` would lead to the following heap:



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	5	**8**	**16**	**10**	**52**	**19**	**25**	**15**	**45**						

You may not create any new data structures. You may not use any other Java classes or methods except native arrays.

```
public class IntMinHeap {
     private int size;  // number of elements in this heap
     private int[] con; // container for values in this heap

/* pre: amount >= 0
   post: if element is present in this heap, it is decreased by amount
     and positioned correctly within the heap. If there are multiple
     copies of element in this heap then the deepest and rightmost
     instance of element is altered.
*/
public void decreaseElement(int element, int amount)
```