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

CS 314 – Final – Fall 2012

Your Name_____

Your UTEID _____

Instructions:

1. There are **5** questions on this test.
2. You have 3 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.
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. When you complete the test show the proctor your UTID. Give them the test and any scratch paper. Please leave the room quietly.

1. (2 points each, 40 points total) Short answer. Place you answers on the attached answer sheets.
   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 returned by the method call `a("TURING")`?

```
public int a(String st) {
     if(st.length() <= 1)
          return st.length() * 2;
     return st.length() + a(st.substring(1)) +
               a(st.substring(2));
}
```

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

```
public int b(int[] data, int w, int h) {
     if(w == h)
          return data[w] % 10;
     else if(w > h)
          return 0;
     else {
          int mid = (w + h) / 2;
          return data[mid] % 10 + b(data, w, mid - 1) +
                    b(data, mid + 1, h);
     }
}
```

C. What is output by the following code?

```
int count = 0;
int toal = 0;
int n = 2000;
for(int i = 0; i < n; i++) {
     for(int j = i; j < n; j++) {
          count++;
          total += count + i * j;
     }
}
System.out.print(count);
```

D. Consider the following method from the `Collections` class in the Java standard library.

`public static int frequency(Collection<?> c, Object o)`
Returns the number of elements in the specified collection equal to the specified object.

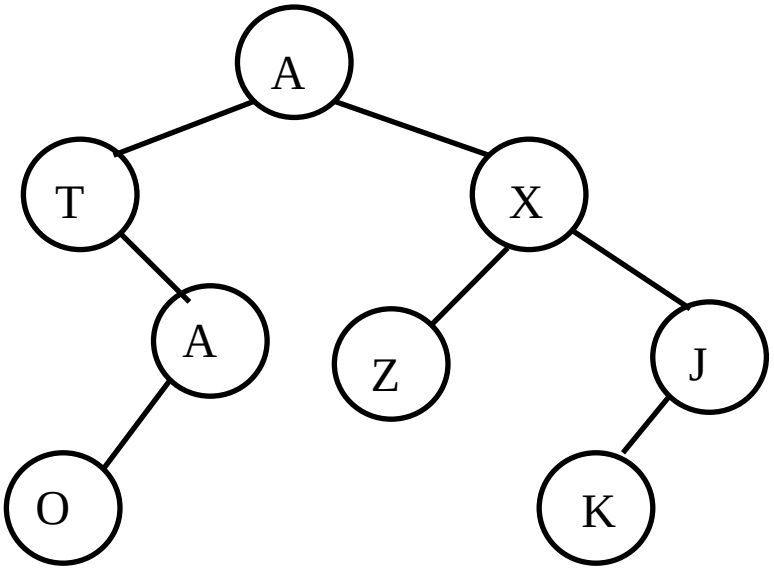
How can the `frequency` method return the correct answer without knowing at compile time what type of `Collection c` is (ArrayList? LinkedList? Hashset? …) and what type of object `o` is? Be specific.

E. The following values are inserted one at a time into a binary search tree using the traditional, naïve algorithm. Draw the resulting tree. `-5, 10, 7, -5, 3, 0`

F. On the answer sheet fill in values for the 5 nodes so the resulting tree is a max heap.

G. What is the worst case order (Big O) to add an element to a hashtable that already contains N elements? The hashtable uses closed address hashing.

For H and I consider the following binary tree:



H. What is the result of a preorder traversal of the tree?

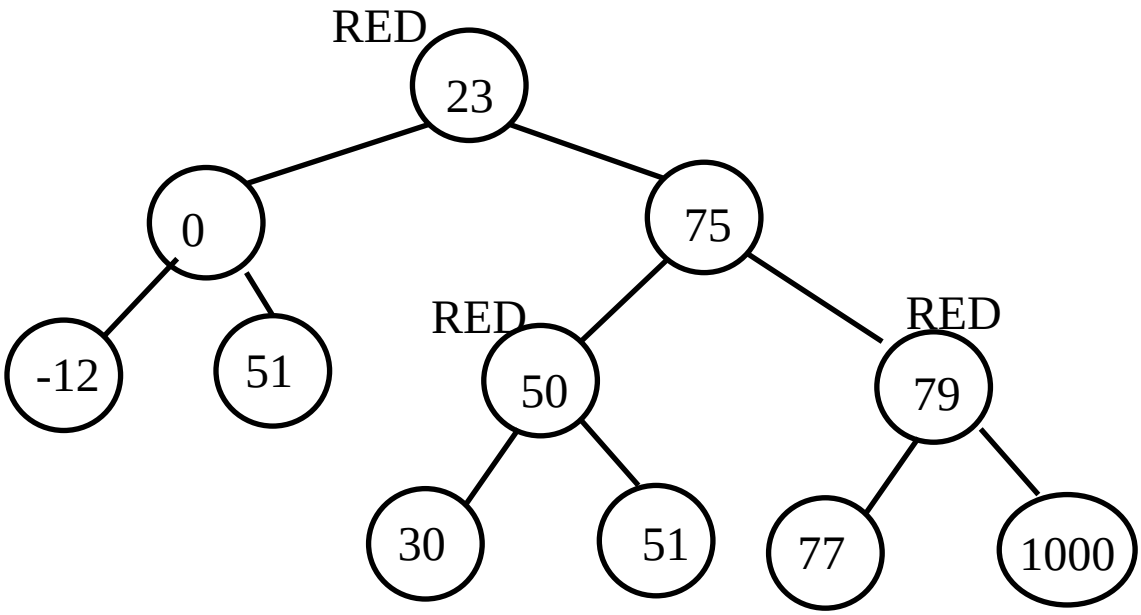
I. What is the result of a postorder traversal of the tree?

J. What is the order (Big O) of method `j` if set is a `TreeSet` from the Java standard library?
What is the order (Big O) of method `j` if set is a `HashSet` from the Java standard library?
N = `num`. Assume the `nextInt` method is O(1).

```
// pre: set != null, set.size() == 0
public void j(Set<Integer> set, int num, Random r) {
     for(int i = 0; i < num; i++) {
          set.add(r.nextInt());
          set.remove(r.nextint());
          set.add(r.nextint());
     }
}
```

K. Suppose you want to encode 60 different names using a fixed length encoding scheme. Represent each name with a unique combination of bits that is the same length as all of the other name codes. No other information about the name is stored. You simply need a unique identifier for each name. What is the minimum number of bits per name required to have a unique code for each name?

J. Consider the following tree. It is not Red-Black tree. Explain all Red-Black tree requirements that are not met. (Nodes not listed as red are black.)



K. N distinct values in ascending order are inserted on at a time into a Red-Black tree. What is the order (Big O) of the height of the resulting tree?

L. What is output by the following client code?

```
ArrayList<Integer> list = new ArrayList<Integer>();
list.add(12);
method_l(list)
System.out.println(list);


public static void method_l(ArrayList<Integer> arl) {
     arl.add(7);
     arl.add(12);
     arl.add(5);
     Collections.sort(arl);
     arl = new ArrayList<Integer>();
     arl.add(9);
     arl.add(-5);
}
```

M. A method that sorts arrays of `int`s uses the mergesort algorithm.

It takes the method 20 seconds to sort an array of 1,000,000 distinct `int`s in descending order.

What is the expected run time when the method sorts an array of 2,000,000 distinct `int`s in descending order?

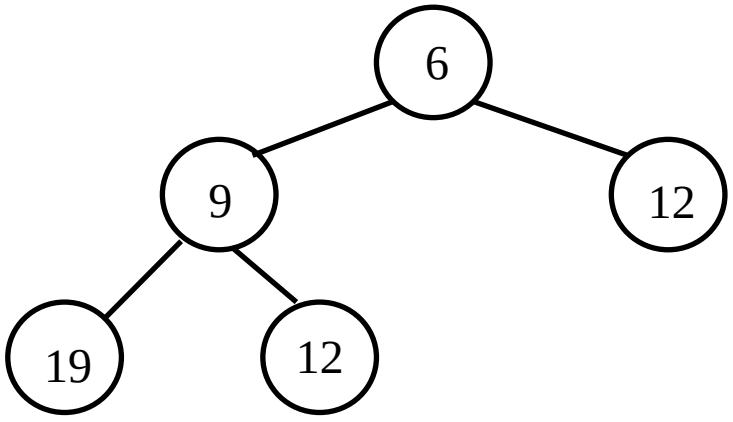
N. Consider the following potential `hashcode` method for an array based list class.

```
public class GenericList<E> {

     private int size;
     private E[] con;
     private int hashcode;

     public int hashcode() {
          if(hashcode == 0) {
               for(int i = 0; i < size; i++) {
                    hashcode += con[i].hashcode();
               }
          }
     }
}
```

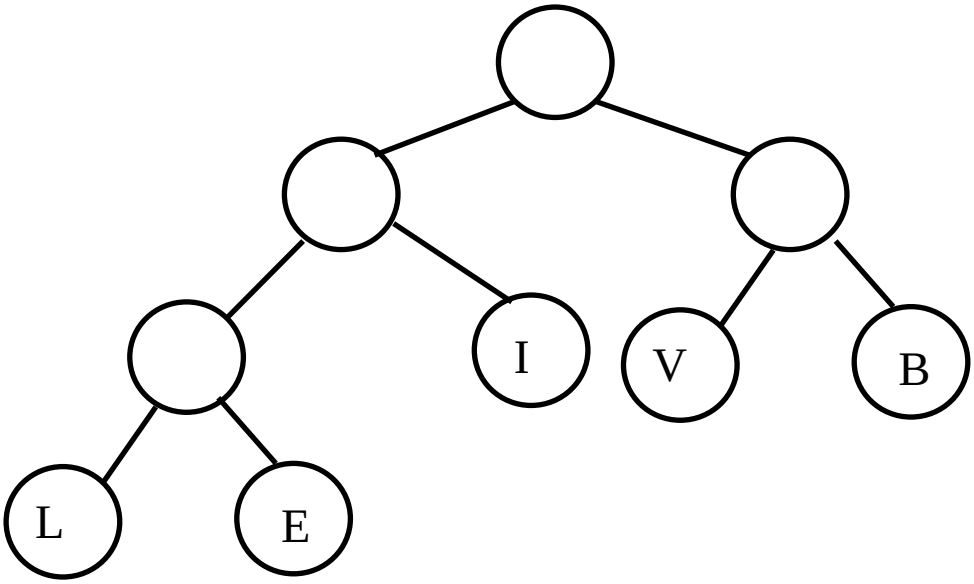
What do you consider the most problematic aspect of the given potential `hashcode` method?

O. In Java, what types of objects may be added to a hash table? Briefly explain why.

P. Consider the following heap that is being used as the internal storage container for a priority queue. Draw the resulting heap if the front element of the queue is dequeued.



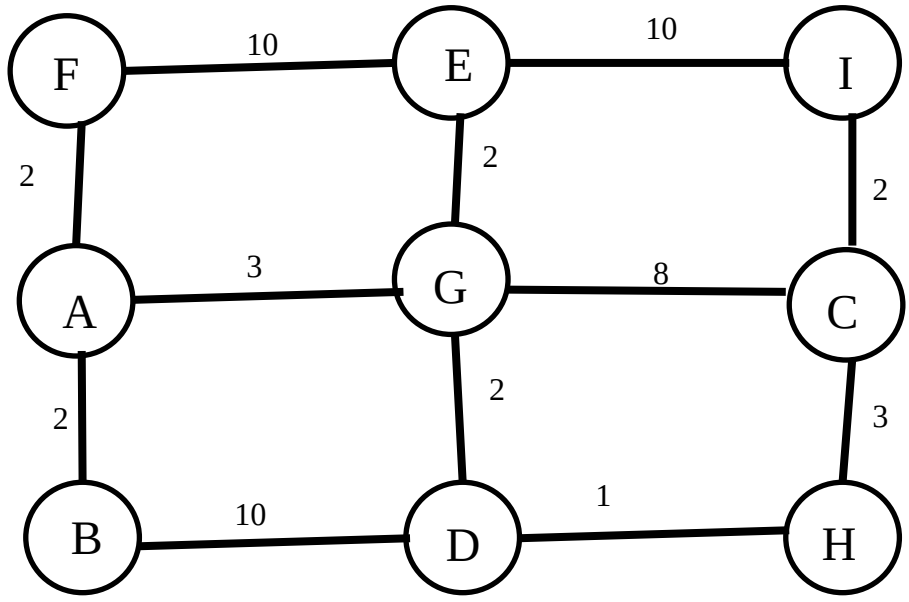
Q. Consider the following Huffman Code tree.



Given the tree above what is the result of encoding `OLIVIA` ? List the correct sequence of 0s and 1s.

R Are most real world graphs sparse or dense? What is the biggest downside of implementing a spares graph with an adjacency matrix?

S. Consider the following undirected, weighted graph. What is the cost of the lowest cost path from vertex B to vertex I?



T. In the Java standard library the `LinkedList` class implements the `Queue` interface but the `ArrayList` class does not? Using Big O, explain why the `ArrayList` class does not implement the `Queue` interface,

EXTRA CREDIT (1 POINT)

Two UTCS Faculty Members have won the Turing Award, the "Nobel Prize" of Computer Science. What are their last names?

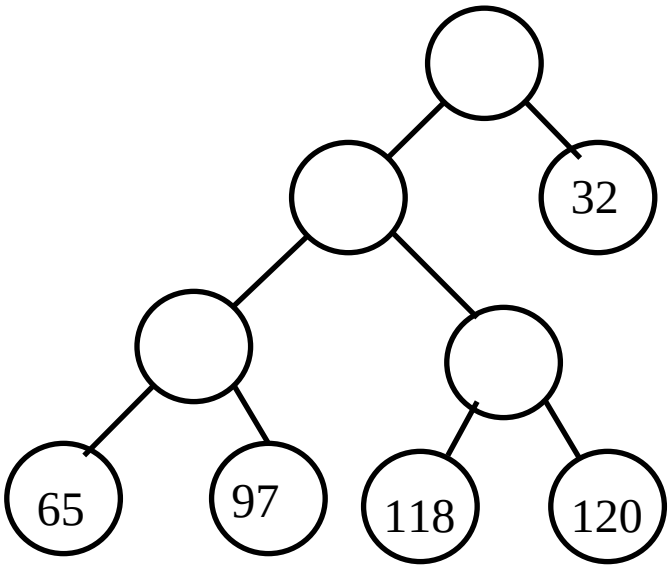
Question 2 starts on the next page.

2. (Huffman Coding - 20 points) This question has 3 parts.

Part A. (12 Points) On the Huffman Coding assignment there were two ways of building the tree from an encoded file. The first was based on the frequency of each chunk. The second was based on a binary representation of the tree itself. This question deals with a third way of building the tree, based on the codes for the chunks themselves. Consider the following example of chunks and codes.

Chunk	Code
65	000
32	1
97	001
118	010
120	011

Given the above chunks and codes the resulting Huffman code tree is:



Complete a private helper method in the Huffman Tree class that adds a new chunk to the current, partially complete tree, given the code for the chunk. Here is the `HuffmanTree` class the method is a part of.

```
public class HuffmanTree {

      private HuffNode root;

      private static class HuffNode {
            // Recall all fields of HuffNode may be accessed by code in
            // the HuffTree class. Recall the class a built in default
            // constructor.

            private int chunk;
            private int freq;
            private HuffNode leftChild;
            private HuffNode rightChild;
      }
```

Complete the following private instance method for the `HuffmanTree` class. The method adds a new chunk to the partial tree based on the chunk's Huffman code, adding nodes and links as necessary to the tree.

The frequency for all nodes (internal and leaf) shall be set to -1.

```
/* pre: root != null, all chars in code are '0' or '1',
          no chunk with code is in this tree yet.
   Add chunk to the tree adding other nodes and links as necessary.
*/
private void addChunk(int chunk, String code) {
     assert root != null
```

2.B (3 points) Complete a constructor for the HuffmanTree class. The constructor accepts a map whose keys are chunks and whose values are Strings which hold the code for a given chunk. The constructor builds a new tree out of the given chunks and codes.

Your method shall call the `addChunk` method from part A. Do not duplicate the code from part A.

```
/* pre: chunksAndCodes != null
        all chars in chunksAndCodes values are equal to '0' or '1'

   post: the HuffmanTree has been created based on chunksAndCodes
*/
public HuffmanTree(Map<Integer, String) chunksAndCodes) {
```

2.C (5 points) Huffman Trees must be complete trees. Recall a complete binary tree is one in which all nodes are leaves or have two children. Write a private instance method that returns `true` if the `HuffmanTree` is complete, `false` otherwise

Your method shall be no worse than O(h) space where h is the height of the tree. In other words you can't use arrays or other data structures equal in length to the number of elements in the tree.

```
private boolean isComplete() {
     assert root != null;
     return completeHelper(root);
}
```

Complete the following method:

```
private boolean completeHelper(HuffNode n) {
```

3. (Graphs - 15 points) Implement a method in a `Graph` class that determines if a given starting node is part of a cycle or not. Recall a cycle exists in a directed graph if there is path with two or more edges starting at a given vertex and eventually reaching that same vertex.

Consider the graph shown in question 1.P on this exam. Vertex E is part of a cycle. One example of a cycle from vertex E is E to G to A to F and back to E.

Recall the Graph class:

```
public class Graph {

    // used to indicate a vertex has not been visited and
    // that no path exists between current start vertex.
    private static final double INFINITY = Double.MAX_VALUE;

    private Map<String, Vertex> vertices;

    private static class Vertex {

        private String name;
        private List<Edge> adjacent;
        private int scratch;
        private double distance;

        public void reset() {
            distance = INFINITY;
            prev = null;
            scratch = 0;
        }
    }

    // model edge between two vertices
    private static class Edge {
        private Vertex dest;
        private double cost;
    }

    // calls the reset method on every vertex in this Graph.
    private void clearAll()

     // returns true if this Graph contains a vertex with the
     // given name.
     public boolean containsVertex(String name)
```

Complete the method on the next page.

Complete the following method.

You may use the Java `Map`, `Iterator`, `Queue`, `LinkedList`, and / or `ArrayList` classes and the `equals` method from the `String` class in addition to the `Edge` and `Vertex` classes. Do not use any other methods from the `Graph` class or `Vertex` class unless you implement them yourself.

```
/* pre: containsVertex(start) == true

   post: return true if the vertex specified by start is part
     of a cycle, false otherwise.
*/
public boolean partOfCycle(String start) {
    assert containsVertex(start);

    clearAll(); // do not change this line of code
```

4. (Hash tables, 10 points) Complete an `Iterator` class for a `Hashtable` class that uses open address hashing. The iterator shall move through the elements of the hash table based on their position in the hash table's internal array. In other words the first element returned by the iterator is the element in the array with the index closest to 0, the next element is the one with the index second closest to 0, and so forth.

Do not use any other methods in the `Hashtable` class unless you implement them yourself.

Your method must be O(1) ***space***, meaning you cannot use temporary arrays, lists, or other data structures whose size depends on the number of elements in the hash table.

Recall the `Hashtable` class:

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

     // number of elements in this Hashtable
     private int size;

     // Elements in con that held a value at one time, but have
     // since been removed refer to EMTPY.
     private static final Object EMPTY = new Object();

     // internal container. Empty elements are either null or
     // refer to the EMPTY object.
     private E[] con;

     public Iterator<E> iterator() {
          return new HashIterator();
     }
```

Complete the following nested class inside the `Hashtable` class. Add instance variables, implement a default constructor, and implement the methods, `hasNext()`, `next()`, and `remove()`. Recall the inner class has access to the private instance variables of the outer class.

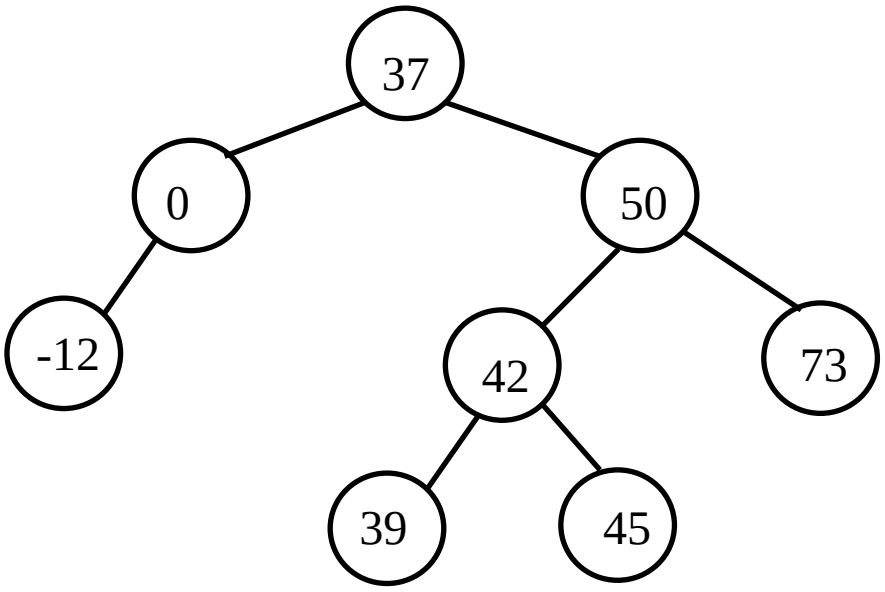
```
     private class HashIterator implements Iterator<E> {

          // add other instance variables here
          private boolean removeOK;
```

```
// complete constructor
private HashIterator() {
    removeOK = false;



// Return true if there are any more elements left in this iteration.
public boolean hasNext() {



// Return the next element in this iteration. (also moves iterator)
public E next() {
     if(!hasNext())
          throw new NoSuchElementException();



// Remove the last element returned by this iterator.
public void remove() {
     if(!removeOK)
          throw new IllegalStateException;
```

5. (Binary Search Trees, 15 points) Implement a method that returns the median value(s) from a binary search tree.

In a set of sorted values the median is the middle element. If there are an odd number of elements the median in the middle element. If there is an even number of elements the median is the average of the two middle elements. Since we are dealing with objects in this question and cannot average them, if the number of elements is even then your method shall return the two middle elements in ascending order.

Consider the following example:



Given the above binary search tree, with an even number of elements your method would return an array of length two with 39 and 42 in it. If the node with 45 were removed there would be an odd number of elements and your method would return an array of length 1 with the value 39 in it.

Your method shall be no worse than O(h) space where h is the height of the tree. In other words you can't use arrays or other data structures equal in length to the number of elements in the tree.

Use an array of length one as a parameter to track the number of nodes that have been visited so far.

Here is the `BinarySearchTree` class:

```
public class BinarySearchTree<E extends Comparable<E>> {

     private BSTNode<E> root;
     private int size;
```

Recall the BSTNode class:

```
public class BSTNode<E extends Comparable<E>> {

     public E getData()
     public BSTNode<E> getLeft()
     public BSTNode<E> getRight()
}
```

This is the kickoff method in the `BinarySearchTree` class:

```
/* pre: size() > 0
   post: return the median value(s) of this BinarySearchTree.
     If the tree size is odd return an array of length 1 with the
     median value. If the tree size is even return an array of length
     two with the two middle values in ascending order.
*/
public E[] getMedian() {
     if(size() == 0)
          throw new IllegalStateException();
     // set result to correct size
     E[] result = (size % 2 == 0) ? (E[]) (new Comparable[2])
                                  : (E[]) (new Comparable[1]);
     int[] count = new int[1];

     medianHelper(root, count, result);
     return result;
}
```

Complete the `medianHelper` method in the `BinarySearchTree` class on the next page.

```
private void medianHelper(BSTNode<E> n, int[] count, E[] result) {
```

Question 1 answer Sheet.

Name_____

A.

_____

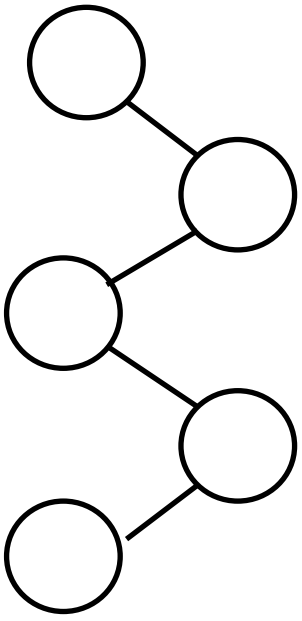
B.

_____

C.

_____

D.



E.

_____

F.

_____

G.

_____

H.

_____

I.

_____

J.

_____

K. Average case order of add method:

Order of add method if load factor near 1.0 and explanation:

_____

L.

_____

M.

_____

N.

_____

1.

O. 2.

_____

NAME_____

P.
_____

Q.
_____

R.
_____

S.
_____

implementation A internal storage container:

implementation B internal storage container:

T.
_____

EXTRA CREDIT : Password for the user id STUDENT is _____