Points off	1	2	3	4				Raw Points Off

Your Name: ____

Your UTEID: ____

Circle your TA's Name: **Anna** **Brad** **David** **Emma** **Justin** **Lilly**
Natalee **Pavan** **Pranav** **Skyler**

Instructions:

1. There are **4** questions on this test. 100 points available. Scores shall be scaled to 250 points.
2. You have 2 hours to complete the test.
3. Place your final answers on this test. Not on the scratch paper. **Answer in pencil**. Exams not completed in pencil are not eligible for a regrade. You may use highlighters on the exam.
4. **This exam shall be entirely your own work.** You may not use **outside resources of any kind** while taking the test. Please remove any smart watches and put them and any mobile devices away.
5. When answering coding questions, ensure you follow the restrictions of the question.
6. Do not write code to check the preconditions.
7. On coding questions 3 and 4 (but not 2) 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 address 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.

1. (2 points each, 50 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 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 is average case O($N^{2)}$, but per the formal definition of Big O it is correct to say Selection Sort is O(N^3) , O(N^4) and so forth.
      Give the most restrictive, correct Big O function. (Closest without going under.)
   e. Assume $\log_2(1,000) = 10$ and $\log_2(1,000,000) = 20$.

A. What is returned by the method call **a(6, 1)**? ______________________

```
public static int a(int x, int y) {
    if (x <= 1) {
        return x + y;
    }
    y++;
    int t = a(x - 2, y);
    return t - y;
}
```

B. Using the techniques and rules from lecture, what is the T(N) of the following method? N = **data.length**. ____________

```
public static int b(int[] data) {
    int r = 0;
    int t = 0;
    for (int i = 0; i < data.length; i++) {
        r += data[i];
        int e1 = data[i];
        for (int j = 0; j < data.length; j++) {
            int e2 = data[j];
            int s = e1 * e2;
            t += s;
        }
    }
    return r - t;
}
```

C. What is the order of the following method? It uses the linked list class we developed in lecture. N = **list.size()**.

____

```
public static double c(LinkedList314<Double> list) {
    double t = 0;
    for (int i = 1; i < list.size(); i *= 2)
        t += list.get(i);
    return t;
}
```

D. What is output when the following code is run? ________________________

```
Object ob1 = new ArrayList<Integer>();
Object ob2 = "CS314";
ob2 = ob1;
System.out.print( ((String) ob2).length());
```

E. What is the worst-case order of the following method? It uses the **java.util.LinkedList** class. N = **list.size()**.

____

```
public static boolean e(LinkedList<Integer> list) {
    for (int i = 0; i < list.size(); i++) {
        int x = list.get(i);
        for (int j = i + 1; j < list.size(); j++)
            if (x == list.get(j))
                return false;
    }
    return true;
}
```

F. What is output by the following code? The stack class is the one developed in lecture and the queue class is a fair, unbuffered queue.

____

```
int[] data = {9, 5, 6, 0, 7};
Stack<Integer> st = new Stack<>();
for (int x : data)
    st.push(x);

Queue<Integer> q = new Queue<>();
while (!st.isEmpty()) {
    int x = st.pop();
    if (x % 3 == 0)
        q.enqueue(x);
}
while (!q.isEmpty())
    System.out.print(q.dequeue() + " ");
```

G. What is the best-case order of the following method? N = **`str.length()`**. ______________

```
public static int g(String str, char tgt) {
    int r = 0;
    final int LIMIT = str.length() / 2;
    for (int i = 0; i < LIMIT; i++) {
        String temp = str.substring(i, LIMIT);
        if (temp.indexOf(tgt) == -1) {
            r++;
        }
    }
    return r;
}
```

H. Consider the following array. It is sent to a sort method. The array is shown at an intermediary step during the sorting algorithm. Of the 5 sorts we discussed in lecture (selection, insertion, radix, quick, merge) which algorithm does the method most likely use?

____

```
[12, 37, 42, 18, 55, 15, 16] // original array

[12, 16, 15, 18, 55, 42, 37] // array at intermediary step
```

I. The following values are added in the order shown, left to right, to an initially empty binary search tree using the simple algorithm demonstrated in class. What is the result of an in-order traversal of the resulting tree?

`2, 0, 0, 0, -3, 5, 5, 2, 2` ____

J. The following values are added one at a time in the order shown into an initially empty red black tree using the algorithm demonstrated in class.
Draw the resulting tree and label the color of all nodes. **1 5 2**

K. The following method takes 40 seconds to complete when **n** = 1,000,000. What is the expected time for the method to complete when **n** = 2,000,000. The method uses the **java.util.TreeSet** class.

____

```
public static TreeSet<Integer> x(int n) {
    TreeSet <Integer> result = new TreeSet<>();
    for (int i = 0; i < n; i++) {
        result.add(i);
    }
    return result;
}
```

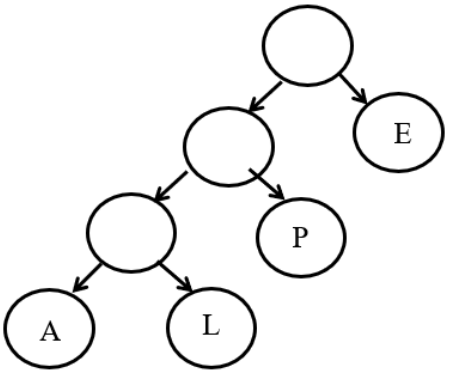
L. For the Huffman file specification from assignment 10 and given **BITS_PER_WORD = 8,** is the following statement, always, sometimes, or never true?

For a given file, the number of bits required for the standard tree format header will be less than the number of bits required for the standard count format header.

____

M. Give the Huffman code tree to the right, what does the following bit sequence decode to?
0 0 0 0 1 0 1 1 0 0 1 0 0 1 1 1

____

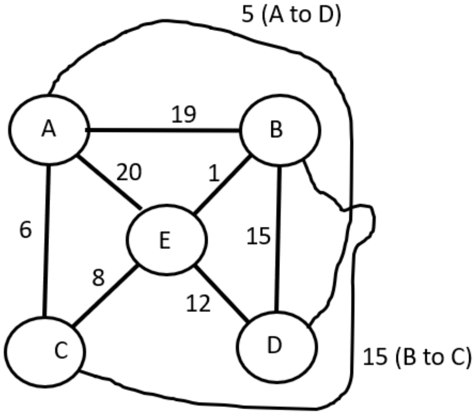


N. Given a graph with V vertices, what is the time complexity to determine the total number of edges in the graph if the internal storage container is an adjacency matrix and if the internal storage container is a **HashMap** of **Vertex** objects with adjacency lists such as our Graph class from assignment 11? In neither case does the graph class maintain an instance variable that stores the number of edges.

adjacency matrix: _________________ hash map of adjacency lists: _________________

O. Consider the graph to the right. We perform Dijkstra's algorithm as demonstrated in lecture to find the shortest paths from vertex A to all other vertices in the graph A is connected to. What order are the vertices marked as visited when Dijkstra's algorithm is performed?

____



Thanks to Kevin Wayne and the Princeton COS 226 course for the inspiration for the following questions.

Consider the following timing data for an unknown graph algorithm given V vertices and E edges. Times are in seconds and are rounded to the nearest hundredth.

	Number at the top of the column is the number of edges in the graph.						
Number at the start of the row is the number of vertices in the graph.		1,000	2,000	4,000	8,000	16,000	32,000
	1,000	1	1.41	2	2.82	4	5.65
	2,000	2	2.82	4	5.65	8	11.31
	4,000	4	5.65	8	11.31	16	22.63
	8,000	8	11.31	16	22.63	32	45.25
	16,000	16	22.63	32	45.25	64	90.51
	32,000	32	45.25	64	90.51	128	181.02

P. What is the expected time for the algorithm to complete given 64,000 vertices and 4,000 edges?

____

Q. What is the expected time for the algorithm to complete given 4,000 vertices and 64,000 edges?

____

R. What is the expected time for the algorithm to complete given 64,000 vertices and 64,000 edges?

____

S. Based on the timing data what is the most likely order of the algorithm in terms of V and E?

____

T. Given a trie data structure as demonstrated in class with a root node that contains the letter c, how many nodes are in the trie (including the root that contains the letter c) if the trie stores the following words?
`cutes, cut, cat, cute, cats, cam, cuts` ____________

U. What is the order of adding a word to a trie given the word has N characters? __________

V. For a min heap is the following statement, always, sometimes, or never true? __________

A level order traversal of the heap results in the values of the heap in ascending order.

W. True or false, hashing with closed addressing, using buckets or chains to resolve collisions, typically requires less memory than hashing with open addressing, using probing to resolve collisions. In each case assume the load limit is 0.75

____

X. The following values are inserted one at a time, left to right in the order shown, to an initially empty min heap using the algorithm demonstrated in class. What is the result of a pre-order traversal of the resulting heap?

```
9  3  9  6  9  5
```

____

Y. Heaps typically use an array as their internal storage container as opposed to connected nodes. Iterating through the elements of the array that are currently present in the heap would be equivalent to what kind of tree traversal?

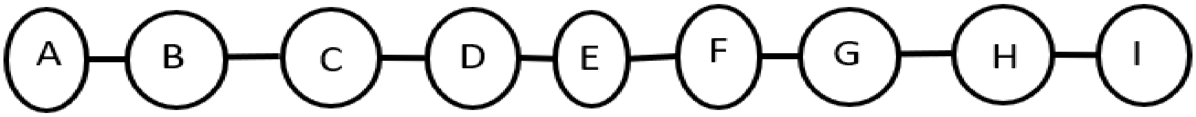
____

Extra Credit: What was your favorite assignment: ____

2. You may want to do **question 3 and / or 4** before this question **Graphs (17 points)** - Complete a recursive backtracking method for the **`Graph`** class from the lectures on graphs and assignment 11 that implements a depth first search to find all vertices that are **exactly** n edges away from a given start vertex. This is referred to as a "n-hop neighborhood".

The paths (sequence of edges) to the vertices that are part of the n-hop neighborhood cannot reuse the same vertex.

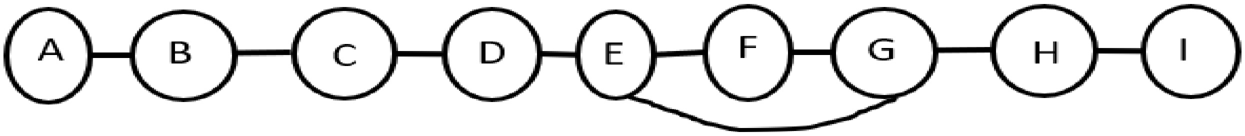
Consider the following, simple example.



If the starting vertex is A and n = 3, then the only other vertex that is 3 edges away is D (A-B-C-D).
Recall, we cannot reuse edges in the path from A to B, B to C, and C back to B is **not allowed** as it reuses the edge from B to C.

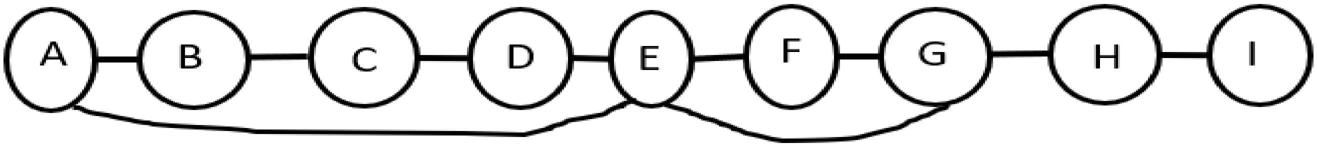
If the starting vertex is E and n = 3 the other vertices 3 edges away are B and H. (E-D-C-B and E-F-G-H)

What if the graph is altered by adding an edge between vertices E and G?



If the starting vertex is E and n = 3 the other vertices 3 edges away are B, H and now I (E-G-H-I). **Note, for this question, a vertex is not part of its own n-hop neighborhood. In the above graph there is a 3-edge path from E to F, F to G, and G to E, but we do not include E in a 3-hop neighborhood of itself.**

What if the graph is altered by adding an edge between vertices E and A?



If the starting vertex is E and n = 3 the other vertices 3 edges away are B, H, I and now C. (E-A-B-C)

Note, we do not include vertices multiple times.

Recall the `Graph`, `Vertex` and `Edge` classes:

```
public class Graph {

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

```
     // Sets scratch to 0 for all Vertex objects in Graph.
     private void clearAll(){ /* ... */ }

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

          private int scratch;
          // For this question no prev or distance variables.
     }

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

// public method that kicks off the recursive helper. Do not alter

     /* start != null, this Graph contains a Vertex with the given
        name.   n >= 2. */
     public ArrayList<String> getNHopNeighbors(String start, int n) {
         ArrayList<String> result = new ArrayList<>();
         clearAll();
         Vertex startVertex = vertices.get(start);
         hopHelp(result, startVertex, n, 0);
         return result;
     }
}
```

Complete the `hopHelp` helper method.

The `Graph` object is not altered other than changing the `scratch` variables of `Vertex` objects.

You may use the provided `Vertex` and `Edge` classes. You may use for-each loops to iterate through `List` of `Edge`s using the for-each loop.

You may use the `add` and `contains` methods for `ArrayList`s.

Do not add any methods or fields to the `Graph`, `Vertex` or `Edge` classes.

Do not add any new helper methods.

Do not use any other Java methods or classes.

Do not create any new objects other than the iterators created implicitly by for-each loops.

Your implementation of the `hopHelp` method must use recursive backtracking.

```
private void hopHelp(ArrayList<String> result, Vertex currentVertex,
                    int goalEdges, int currentEdges) {
```

3. (**Trees, 16 points**) This question involves binary trees that store ints, `IntTree`s. The trees are NOT binary search trees. Write an instance method for the `IntTree` that determines the sum of values in nodes in the tree that contain a value with a given range **and** are at a depth within a given range.

Consider this example. If the value range is [4, 6] and the depth range is [1, 2] then the method would return 9. The only nodes with a value between [4, 6] inclusive and at a depth between [1, 2] inclusive are the nodes that contain 4 and 5. 4 + 5 = 9

```
calling IntTree
     3
   /   \
  4     7
 / \   / \
0   9  2  5
   / \
  8   6
```

The method header:

```
public int inRangeSum(int[] valueRange, int[] depthRange)
```

The `int` arrays `valueRange` and `depthRange` shall both have a length of 2. The first element indicates the start of the range (inclusive) and the second element indicates the end of the range (inclusive). The first element shall in the array shall always be less than or equal to the second element.

Note, neither valueRange or depthRange are altered as a result of this method.

The `IntTree` and nested `IntNode` classes:

```
public class IntTree {
    private IntNode root; // null iff size == 0

    // to be completed
    public int inRangeSum(int[] valueRange, int[] depthRange)

    private static class IntNode {
        private int val;
        private IntNode left;  // null if left child doesn't exist.
        private IntNode right; // null if right child doesn't exist.
    }
}
```

Do not use any other Java classes or methods besides the given `IntTree` and `IntNode` classes.

Do not create any new objects. No new arrays. No new `IntNode` objects.

You can, of course have refences to existing `IntNode` objects.

```
/* pre: per the problem description
    post: neither valueRange or depthRange are altered as a result of
        this method and per the problem description. */
public int inRangeSum(int[] valueRange, int[] depthRange)
```

4. **Hash tables and linked lists** (17 points) Complete the **add** method for a hash table class. The class uses closed addressing. For its buckets (or chains) it uses a linked structure of nodes based on a nested node class. The hash table's internal array stores **null** if no elements are in the bucket at that index. If there are one or more elements in that bucket the array element refers to the first node in the structure. The length of a given bucket is not tracked, nor is there a dedicated reference to the last node in a bucket. Just the reference to the first node in the structure, if the bucket isn't empty. The last node in the structure has its next reference set to **null**. The **Object**s (data elements) in a chain of **Node**s are not in any particular order.

Recall a given object is stored at most once in a given hash table. Duplicates are not allowed.

Recall if the load factor of hash table (number of elements stored / length of array) is greater than the load limit for the hash table, the array is resized and elements are rehashed. Assume a private **resize** method is completed and available for your use.

Finally, the hash table is generic based on the **Object** class, inheritance and polymorphism, not the Java Generic Type syntax. (Poor clients.)

The **HashTable** class for this question:

```
public class HashTable {

     // The internal array of buckets. This never equals null.
     private Node[] con;
     private int size; // Number of elements in this HashTable.
     private final double LOAD_LIMIT; // Set in constructor.

     public boolean add(Object o) // to be completed

     private void resize() // Already done and your code may call.

     private static class Node {
         private Object data;
         private Node next; // null if I am the last Node in a bucket.
         public Node(Object d, Node n) {data = d; next = n;}
     }
}
```

Complete the add method for this hash table class. The method returns **true** if the hash table was altered by the call to add, **false** otherwise.

You may use the given hash table and **Node** classes, including the **resize** method.
You may use the following Java methods.

The **int hashCode()** method on **Object**s.
The **boolean equals(Object o)** method on **Object**s.
The **Math.abs(int x)** method that returns an **int**.

You may create one new Node object if necessary.
You may, of course, have declare and use references to existing objects.

Do NOT use recursion in your answer.

```
// pre: o != null, post: per the problem description
public boolean add(Object o) {
```