Points off	1	2	3	4			Raw Points Off

Your Name: ____

Your UTEID: ____

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 silence your mobile devices**. 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 2 and 3, you may implement your own helper methods. Not 4.
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.
10. When you complete the test show the proctor your UTID and give them the test. Please place used and unused scratch paper in the appropriate boxes at the front of the room. Please leave the room quietly.

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 `log₂(1,000) = 10` and `log₂(1,000,000) = 20.`

A. What is output when the method call **`a(9, 1)`** is made?

____

```
public static void a(int x, int y) {
    if (x < y)
        System.out.print(x - y);
    else {
        System.out.print(x);
        a(x / 2, y + 1);
        System.out.print(y);
    }
}
```

B. Using the techniques and rules from lecture, what is the T(N) of the following method? N = the parameter **n**. Assume **n** is a power of 2. In other words, **log₂n** shall equal a non-negative integer.

____

```
public static double b(int n) {
    int t = 0;
    for (int i = 1; i <= n; i *= 2) {
        for (int j = 0; j < n; j++) {
            t += i * j;
        }
    }
    return t;
}
```

C. The following **char**s are inserted one a time in the order shown, left to right, into an initially empty binary search tree using the simple add algorithm presented in lecture. What is the result of a post order traversal of the resulting tree? Do not include the quotes on the **char**s in your answer.

'N' 'I' 'D' 'H' 'I' _______________________________

D. The following code uses the **BST314** class developed in lecture. The **iterativeAdd** method implements the simple add algorithm demonstrated in lecture, iteratively, without recursion. The method takes 5 seconds to complete when **n** = 40,000. What is the expected time for the method to complete when **n** = 80,000?

____

```
public static BST314<Integer> d(int n) {
    BST314<Integer> t = new BST314<>();
    for (int i = n; i >= -n; i--) {
        t.iterativeAdd(i);
    }
    return t;
}
```

E. The following method uses the Java **TreeSet** class. The method takes 5 seconds to complete when **n** = 1,000,000. What is the expected time for the method to complete when **n** = 4,000,000?

____

```
public static TreeSet<Integer> e(int n) {
    TreeSet<Integer> t = new TreeSet<>();
    for (int i = n; i >= -n; i--) {
        t.add(i);
    }
    return t;
}
```

F. The following `int`s are inserted one a time in the order shown, left to right, into an initially empty red black tree using the algorithm presented in lecture. Draw the resulting tree labeling each node red or black. *(Inspired by Gracelynn)*

`9   7   5   3`

G. The following array of elements is sorted using a base 10 radix sort as demonstrated in class. What are the contents of the array after the completion of the second pass of the radix sort?

`[725, 124, 131, 99, 1003]`

____

H. The following text, just the English letters show, no whitespace or any non-letter characters, are placed into a Huffman Code tree of characters as demonstrated in lecture. A pseudo end of file value is **NOT** added to the tree. What is the result of a level order traversal of the resulting tree? The level order traversal includes only leaf nodes. The level order traversal ignores internal nodes. *(Inspired by Pavan, of course.)*

`PAVANAN` ____________________________________

I. How many bits would be necessary to encode (flatten) a Huffman code tree from assignment 10, using the *Standard Tree Format* (also from assignment 10) with 10 **internal** nodes? **Do not** include the 32 bits for the size of the tree that are written to the header before the bits that actually describe the tree, only the bits needed to represent the tree using the specification from assignment 10. *(Inspired by Gracelynn and Sebastian)*

____

J. For Huffman coding to result in fewer bits to represent a file that includes all possible values given a **`BITS_PER_WORD`** equal to 8 which of the following trees is desired? (State the letter of the single best answer.)

A. A tall, skinny tree __________________
B. A short, board tree
C. Either will likely result in compression

K. At the point marked **`Point K`** in the following code what is the size of the stack **`st`**? Use Big O notation. N = the parameter **`n`**. The **`Stack314`** class is the one we developed in lecture. *(Inspired by a Princeton COS 226 question.)*

____

```
public static void k(int n) {
    n = Math.abs(n);
    Stack314<Integer> st = new Stack314<>();
    while (n > 2) {
        st.push(n % 3);
        n /= 3;
    }
    st.push(n);
    // POINT K
    while (!st.isEmpty()) {
        System.out.print(st.pop());
    }
}
```

L. If the goal of method **`k`** is to print out the base 3 representation of the absolute value of the parameter **`n`**, does the method accomplish that goal? For example, given **`n = 27`** the goal of the code is to output **`1000`**

A. No
B. Yes

____

M. What is the order of the following code? N = **`data.length`**. The elements in data are initially in random order. The code uses the **`java.util.PriorityQueue`** class.

____

```
public static void m(int[] data) {
    PriorityQueue<Integer> pq = new PriorityQueue<>();
    for (int x : data) {
        pq.add(x); // enqueue operation
    }
    for (int i = 0; i < data.length; i++) {
        data[i] = pq.remove(); // dequeue operation
    }
}
```

N. What **single word** best describes the properties of the elements in the array **`data`** at the completion of method **`m`**?

____

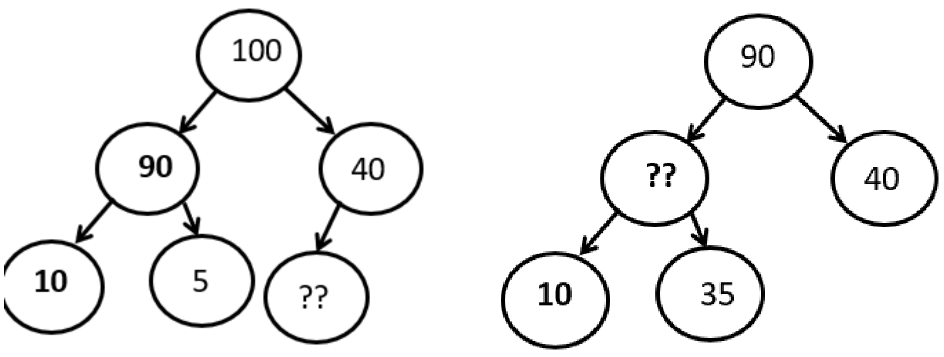
O. The following values are inserted one at a time in the order shown, left to right, into an initially empty min heap using the algorithm demonstrated in class. How many swaps of values occur in total while adding the values shown? *(By Namish)*

```
5  15  20  5  0  25
```

____

P. A **`removeMax()`** operation is performed on the ***maximum binary heap*** shown on the left producing the maximum binary heap on the right. The heap stores integers. Which of the following values could the ?? actually be? List all valid answers. *(Inspired by a Princeton COS 226 question.)*

____



A. 30
B. 37
C. 40
D. 47
E. 55

Q. Assume a **`Point`** class for representing points on a Cartesian plane has instance variables **`x`** and **`y`**, both **`int`**s. The **`hashCode`** for the **`Point`** class is as follows.

```
public int hashCode() {
    return x + y;
}
```

The following **`Point`** objects are inserted in the order shown into a hash table with an internal array of length 5 and a load limit of 0.75. The hash table resolves collision with linear probing as demonstrated in class. What are the contents of the internal array after adding the **`Point`** objects. Use forward slash to indicate array elements that store **`null`**. *(Inspired by Bersam)*

```
(4, 10), (0, 4), (3, 4)
```

0	1	2	3	4

R. What is the most likely output for the following method when **n** = 1,000,000? The method uses the **java.util.HashSet** class.

____

```
public static void r(int n, Random rng) {
    HashSet<Integer> hs = new HashSet<>();
    for (int i = 0; i < n; i++) {
        int x = Math.abs(rng.nextInt()); // Line 1
        x = x % 10; // Line 2
        hs.add(x); // Line 3
    }
    System.out.println(hs.size());
}
```

S. Method **r** shown above takes 5 seconds to complete when **n** = 1,000,000. What is the expected time for the method to complete when **n** = 2,000,000? Assume the **Random.nextInt()** and the **println** methods are O(1).

____

T. The lines marked **Line** 1 and **Line 2** in method **r** are deleted. The line marked **// Line 3** is altered as shown below. Assume the default constructor in the **HashSet** class creates an array with 10 elements and sets the load limit of the internal hash table to 0.75. When the hash table resizes its internal array, it doubles the capacity of the array. How many times will the hash table likely resize its internal array if **r** is called with the parameter **n** = 90?

____

```
        hs.add(rng.nextInt()); // New Line 3
```

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 second.

	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.		10,000	20,000	40,000	80,000	160,000
	10,000	1	4	16	64	256
	20,000	2	8	32	128	512
	40,000	4	16	64	256	1000
	80,000	8	32	128	512	2000
	160,000	16	64	256	1000	4000

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

____

V. When we demonstrated the **`Trie`** class in lecture we added all the words of an English dictionary, with roughly 113,000 words, to the **`Trie`**. The sum of the lengths of the words in the dictionary (total number of characters) was on the order of 900,000. Approximately how many nodes were in the resulting **`Trie`**? Pick the single best answer.

____

A. 230,000 B. 460,000 C. 700,000 D. 900,000 E 1,800,000

W. This question involves a directed, **unweighted** graph. The graph is represented below with an adjacency list. The first letter in a line is the source vertex name. This is followed by the vertices adjacent to the souce vertex. For example, there is an edge from vertex A to vertex B, but no edge from vertex B to vertex A. Is the given graph cyclical, yes or no?

____

A: B, C
B: D, G
C: B, D
D: E
E: A, C, D, F
G: H

X. Given the graph from question 1.W what is the length (number of edges) of the shortest path **from** vertex G **to** vertex A?

____

Y. What is output by the following code? ____

```
 int[] result =
         IntStream.range(1, 14)
         .filter(x -> x % 2 == 0 && x % 3 == 0)
         .map(x -> x * 2)
         .toArray();
System.out.print(Arrays.toString(result));
```

Extra Credit (2 points): What was your favorite programming assignment in CS314? (number or name)

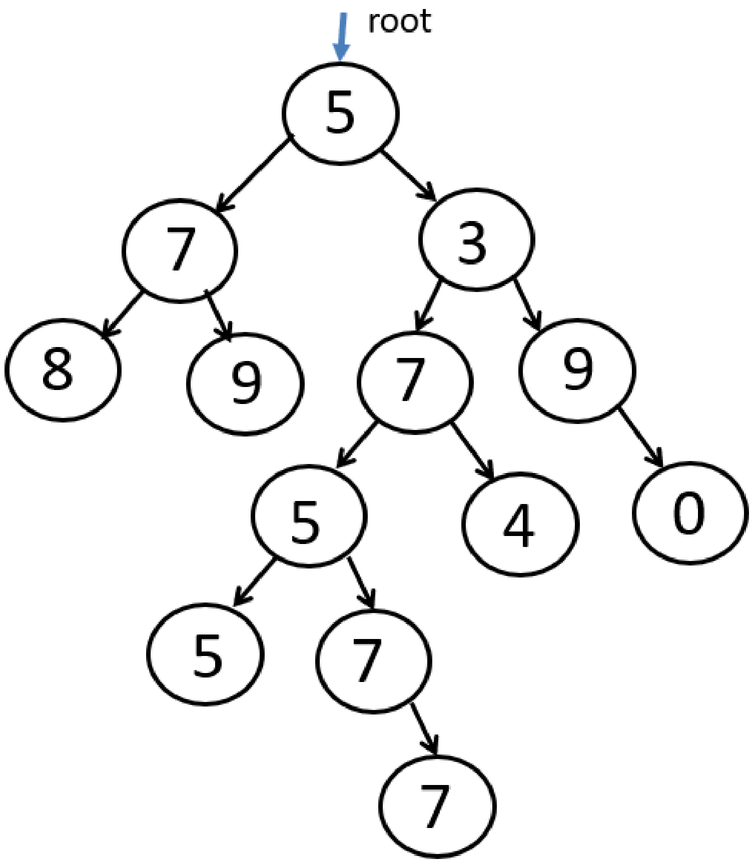
____

2. **Trees (15 points)** - Complete an instance method for a binary tree class that returns the number of nodes that meet the following conditions:

1. The node contains a value equal to a given target value.
2. The depth of the node is greater than or equal to a given depth.
3. The node has exactly two children.

```
public int count(E tgt, int minDepth) {
```

Consider this example with a `BinTree` that contains `Integer` objects. The root contains the value `5`. Assume the variable `t` refers to the tree to the right.



```
t.count(7, 6) -> returns 0
t.count(7, 5) -> returns 0
t.count(7, 4) -> returns 0
t.count(7, 3) -> returns 0
t.count(7, 2) -> returns 1
t.count(7, 1) -> returns 2
t.count(7, 0) -> returns 2
t.count(5, 0) -> returns 2
t.count(3, 1) -> returns 1
t.count(3, 2) -> returns 0
```

The `BinTree` and nested `BNode` classes:

```
public class BinTree<E> {
    // stores null iff tree is empty
    private BNode<E> root;

    public int count(E tgt, int minDepth) // to be completed

    private static class BNode<E> {
        private E data; // Never null.
        private BNode<E> left;  // null if left child doesn't exist.
        private BNode<E> right; // null if right child doesn't exist.
    }
}
```

You may use the `Object.equals` method and the given `BinTree` and `BNode` classes.

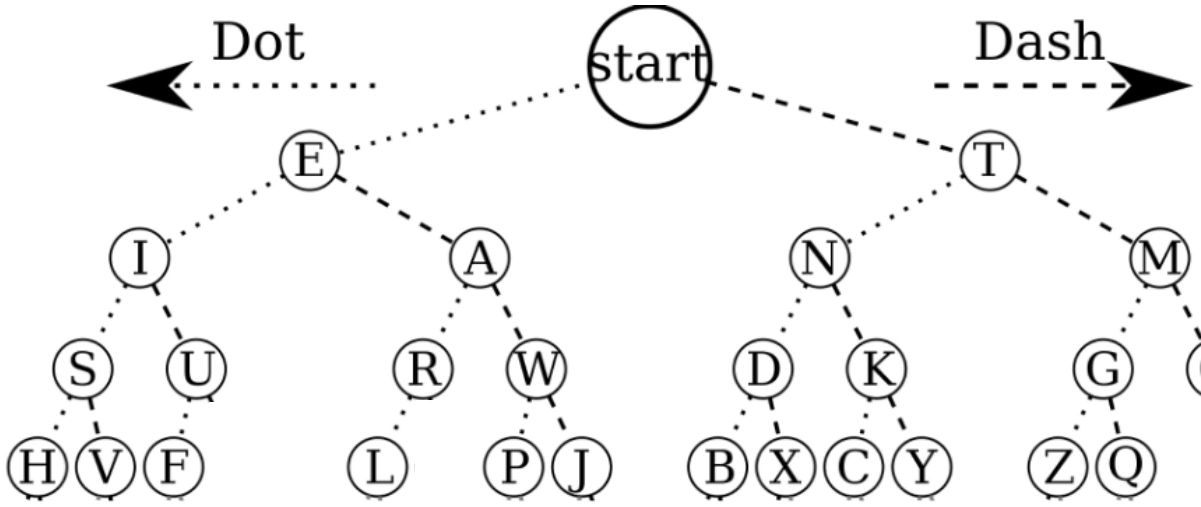
Do not use any other Java classes or methods besides those.

Do not create any new objects, not even arrays of length 1.

```
/* pre: tgt != null, minDepth >= 0
   post: Per the problem description. This BinTree is not altered
         by this method. */
public int count(E tgt, int minDepth) {
```

3. **Encoding and Trees (19 points)** *(By Nidhi)* As mentioned in the lectures on Huffman encoding, Morse code is an encoding scheme for encoding English letters and other characters. Unlike Huffman encoding, the encoding for Morse code is fixed and some codes ***are*** the prefix for other codes.

Consider this tree which shows the standard Morse code encodings for English letters. (Note, do not assume the **`MorseCodeTree`** for this question is exactly the same. You **may** assume every valid path in the tree ends in a node with a valid symbol.)



Assume a period, '.' represents dot and a hyphen, '-', represents dash. The Morse encoding for E is . The Morse encoding for A is .- Notice the code for E is a prefix for the code for A. Thus, unlike Huffman we need a separator symbol. We will use an asterisk, '*' as our separator symbol. Every encoded letter, even the last one, shall be followed by a * in a properly encoded message.

Complete a method in a **`MorseCodeTree`** class that accepts a **`String`** with the encoded message and returns a **`String`** with the decoded message: **`public String decode(String s)`**

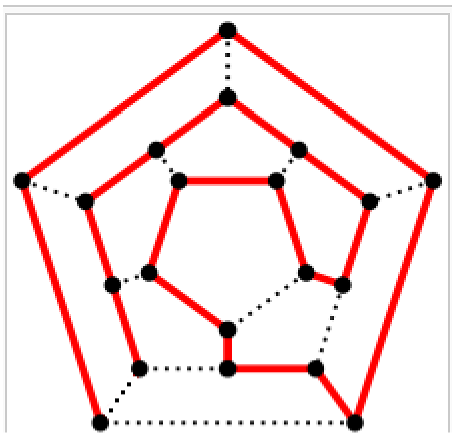
Consider these calls to decode and the expected results. Assume **`t`** is a **`MorseCodeTree`** object.

```
t.decode("-.*..*-..*....*..*") -> returns "NIDHI"

t.decode("..-*-*-.-.*...*") -> returns "UTCS"

t.decode("-.-.*.-*...*.*-.--*") -> returns "CASEY"

t.decode("..--*-*") -> returns null, no code with ..--

t.decode("..--*-") -> returns null, no ending asterisk.

t.decode("*.*.*" -> returns null, cannot start with "*".

t.decode("..*+-*") -> returns null, invalid symbol: +

public class MorseCodeTree {
     private MNode root; /* Root of the Morse code tree.
                            Does not store a valid letter. */

     private static class MNode {
         private char letter;
         private MNode left; // null if no left child
         private MNode right; // null if no right child
```

Complete the following instance method for the **`MorseCodeTree`** class. You may use the **`root`** instance variable and the instance variables of **`MNode`** objects. You may use the **`String`** **`charAt`** and **`length`** methods. You may create a new empty **`String`** and you may create new **`String`**s via concatenation. (For this question, we are not worrying about the inefficiency of String concatenation.) **Do not use any other Java classes or methods. Do not create any new data structures. Do not use recursion.**

```
/* pre: s != null, s.length() >= 1
   post: return the message encoded by s or null if s is not a valid
   encoding. This MorseCodeTree is not altered by this method call. */
public String decode(String s) {
```

4. **Graphs and Recursion (16 points)** Implement a recursive backtracking helper method for an instance method for the **`Graph`** class from assignment 11 that determines if the **`Graph`** contains a ***Hamiltonian Path*** or not. It is NOT necessary to store the Hamilton path in any way.



A Hamilton Path is a path through a graph where every vertex in the graph is visited **exactly** once. Note: It is **not** necessary to use every edge in the graph. The start and end vertices are different.

Consider the Hamiltonian path in the graph to the right. Used edges are solid and unused edges are dashed.

Recall the **`Graph`** class from assignment 11.

```
public class Graph {

    private Map<String, Vertex> vertices;
    private String currentStartVertex;

    private void clearAll() /* Sets all Vertex object's scratch to 0. */

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

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

You may use the following methods: **`Map -> (get and size), List -> (get, size, iterator) Iterator -> (hasNext and next`** explicitly or implicitly (for-each loops)). You may use the **`Graph`**, **`Vertex`**, and **`Edge`** instance variables. Do not use any other methods from the **`Graph`** class or **`Vertex`** class.

Do not create any new data structures other than iterators.

Do not add any other helper methods or alter the parameter list of the helper.

```
public boolean containsHamiltonianPath() {
    clearAll();
    for (String name : vertices.keySet()) {
        if (helper(name, 0)) {
            currentStartVertex = name;
            return true;
        }
    }
    return false;
}
```

```
/* Complete the following method. Do not add any other helper methods.
   DO not change the method header. Do not add or remove elements from
   the map named vertices. */
private boolean helper(String currentVertexName, int verticesInPath) {
```