Points off	1	2	3	4		Total off	Net Score

Your Name: _____

Your UTEID: _____

Circle your TA's Name: **David K. David T. Elizabeth Henry Lilly Nina Pranav Skyler Sam Trisha**

1. There are **4** questions on this test. 100 points available. Scores will 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 completed with pen are not eligible for a regrade.**
4. You may not use a calculator or **outside resources of any kind** 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.
7. On coding questions, you may implement your own helper methods **except question 2.**
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. (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. The following method takes 4 seconds to run when n = 1,000,000. **What is the expected time for the method to complete when n = 3,000,000?** Assume the **`Random`** constructor and **`nextInt()`** methods are O(1). _____

```
public static List<Integer> getList2(int n) {
    Random r = new Random();
    ArrayList<Integer> result = new ArrayList<>();
    for (int i = 0; i < n; i++) {
        int x = r.nextInt();
        if (x % 3 == 0)
            result.add(0, x); // position, val to add;
        else
            result.add(x);
    }
    return result;
}
```

B. The following method takes 20 seconds to complete when n = 1,000,000. **What is the expected time for the method to complete when n = 2,000,000?** The binary search tree class use the simple add method demonstrated in class and is implemented iteratively.

```
public static BinarySearchTree<Integer> make(int n)
    BinarySearchTree<Integer> result;
    result = new BinarySearchTree<>();
    for (int i = 0; i < n; i++) {
        result.add(i * 10);
    }
    return result;
}
```

_____

C. The following method takes 1 second to complete when n = 1,000,000. **What is the expected time for the method to complete when n = 4,000,000?** The binary search tree class use the simple add method demonstrated in class. Assume the **`Random`** constructor and **`nextInt()`** methods are O(1).

```
public static BinarySearchTree<Integer> make2(int n)
    BinarySearchTree<Integer> result;
    result = new BinarySearchTree<>();
    Random r = new Random();
    for (int i = 0; i < n; i++) {
        int val = r.nextInt() % 10;
        result.add(val);
    }
    return result;
}
```

_____

D. The following values are inserted one at time in the order show, left to right, into an initially empty binary search tree using the simple insertion algorithm demonstrated in class. **(NOT the Red - Black tree insertion algorithm)**

What is the depth of the the node that contains the value 5 in the resulting tree? _____

`VALUES: 17  30  17  -10  10  12  0  30  5  21  -10  10  5  -5`

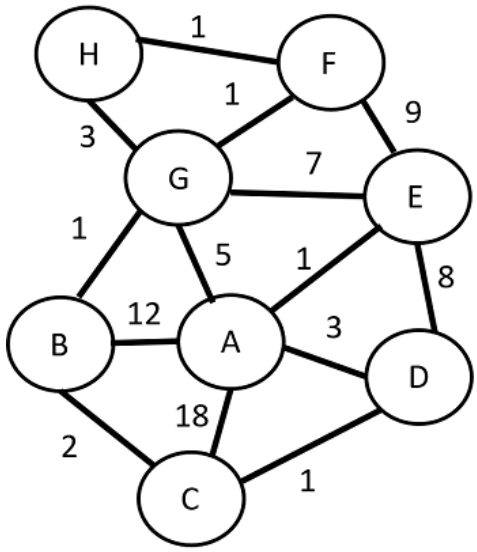
E. Suppose we are creating a new file format to represent information about college students. Each student record must store the college or university the student is currently attending. Assume students can attend exactly one college or university at a time. A student cannot attend multiple colleges or universities at the same time.

There are 3,000 possible college and universities a student could be attending. Each college and university has a distinct name.

If our goal is to minimize the number of bits in individual student record files used to represent the college or university a student is attending, what is the minimum number of bits necessary to encode the 3,000 possible college and universities in the file?

_____

F. Consider the following undirected graph to the right. We apply Dijkstra's algorithm as demonstrated in class, use H as the start vertex, and find all the shortest path from H to every other vertex in the graph.



Which of the following shows the priority queue of `Path` objects at an intermediary step in the process?

The front of the queue is on the left and the back of the queue is on the right.

A. [[F, 1], [G, 2], [B, 3], [G, 3], [F, 4], [E, 9]]
B. [[F, 1], [E, 10], [A. 9], [A, 11]]
C. [[G, 3], [F, 4], [B, 4], [C, 5]]
D. [[G, 2], [G, 3], [E, 10]]
F. None of the answers shown are possible.

_____

G. **Which of the following statements about real world graphs are true?** _____

A. Most real-world graphs are sparse.
B. The number of real-world graphs that are sparse is roughly equal to the number of real-world graphs that are dense.
C. Most real-world graphs are dense.

H. What is the maximum height of a Huffman Code Tree with 19 total nodes? _____

I. Assume the **`hashCode`** method for the **`Integer`** class simply returns the corresponding int value for the **`Integer`** object. (**`hashCode`** method for **`Integer`** 3 returns **`int`** 3.)

The **`Integers`** below are inserted into a hash table in the order shown below, left to right. The hash table uses open addressing and linear probing to resolve collisions as demonstrated in lecture. The internal array initially has a length of 10. **`null`** elements are shown with a back slash, \.

```
22, 112, 999, 10, 112, 13, -3, 0
```

Which of the arrays below correctly shows the elements of the array after the eight Integers are added to the hash table in the order shown, left to right.

A. `[10, \, 22, 13, \, \, \, \, \, 999]`
B. `[10, 0, 22, 112, 13, \, \, \, \, 999]`
C. `[10, 0, 22, 112, 112, 13, -3, \, \, 999]`
D. `[0, 10, 22, 13, -3, 112, \, \, \, 999]`
E. None of the answers A - D are correct.

_____

J. Using the techniques developed in lecture, what is the T(N) (actual number of executable statements given N data elements) of the following method? N = **`data.length`**

_____

```
public static int j(int[] data) {
    int result = 0;
    for (int i = 0; i < data.length; i++) {
        if (i % 2 == 0) {
            int t = data[i];
            result += t;
        } else {
            for (int j = 0; j < data.length; j++) {
                int x = data[i];
                int y = data[j];
                int z = x * y;
                result += z;
            }
        }
    }
    return result;
}
```

K. The following values are added to a min heap in the order shown, left to right, using the algorithm demonstrated in lecture. Draw the resulting min heap.

`12, 5, 5, 12, 8, 5`

L. What is the order (Big O) of the **`methodL`** method shown below? The **`HashSet`** class is the built in Java **`HashSet`** class. Assume the **`nextInt`** method is O(1). N = the parameter n.

_____

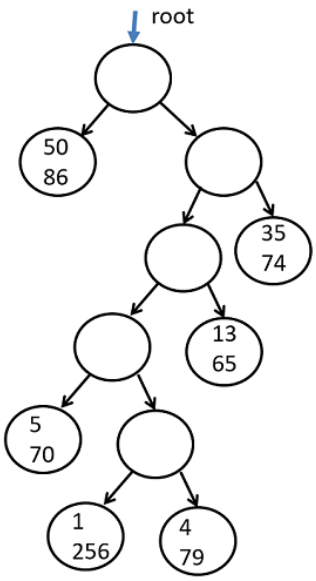
```
public static Set<Integer> methodL(Random r, int n) {
    Set<Integer> result = new HashSet<>();
    for (int i = 0; i < n; i++) {
        result.add(r.nextInt(i + 1));
    }
    return result;
}
```

M. The following method takes 1 second to complete when n = 10,000, 4 seconds to complete when n = 20,000, and 16 seconds to complete when n = 40,000. Of the sorting algorithms presented in class (selection sort, insertion sort, radix sort, quick sort, and mergesort) which one(s) could be used by the **sortM** method given the above time data? List all that are reasonably possible.

_____

```
public static int[] methodM(int n) {
    int[] ar = new int[n];
    sortM(ar);
    return ar;
}
```

N. Consider the Huffman Code tree to the right. In the leaves, the top number is the frequency of the value in the original file and the bottom number is the value from the original file.

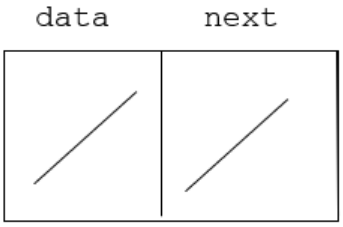


Using the Standard Tree Format (STF) from assignment 10, how many bits are needed to encode the tree?

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 itself.

_____

O. Consider a singly linked node class like the one used in lecture when we developed our singly linked list. This **Node** class does not use Java generics. Each node has a reference to an object and another **Node**.



Consider the following code. The next reference in each **Node** is a public variable of type **Node**. The data reference in each **Node** is a public variable of type **Object**. The zero-argument constructor sets the data and next instance variables to **null**.

Draw the variables, references, and objects that exist after the following code executes. Draw node objects as shown above and boxes for variables. The example has all instance variables set to **null**.

```
Node n1 = new Node();
Node n2 = new Node();
Node n3 = n1;
n1.data = "CS";
n2.next = n3;
```

P. Consider the class named **`Point2`** to the right. The **`Point2`** class has no code other than what is shown.

```
public class Point2 {
    public int x;
    public int y;

    public Point2(int x, int y) {
        this.x = x; this.y = y;
    }

    public int hashCode() {
        return x * 37 + y * 73;
    }
}
```

What is output by the following code? The HashSet class is the standard Java HashSet class.

```
HashSet<Point2> hs = new HashSet<>();
Point2 p1 = new Point2(12, 12);
hs.add(p1);
hs.add(new Point2(5, -5));
Point2 p2 = new Point2(5, -5);
System.out.print(hs.contains(p2) + " ");
System.out.print(hs.contains(p1));
```

_____

Q. The following values are inserted in the order shown, left to right, into an initially empty binary search tree using the simple insertion algorithm shown in class. What is the result of post order traversal of the resulting tree?

`7, 17, 12, -5, 0, 12, 17, 3, -5` _____

R. Draw a Trie that contains the following words. **Place s in the root node of the Trie.**

`seed, seek, seal, seem, seel, sea`

Use a check in the nodes that terminate in a word and a - in the nodes that do not terminate in a word.

S. What is the order of the following method? _____

N = `list.size()`. The question uses the Java **`LinkedList`** class.

```
// pre: list.size() >= 10
public static int s(LinkedList<Integer> list) {
    int result = 0;
    for (int i = 0; i < list.size(); i++)
        result += list.get(i % 10);
    return result;
}
```

T. What is the order of the following method? _____

N = the parameter **n.**

```
public static ArrayList<Integer> t(int n) {
    ArrayList<Integer> list = new ArrayList<>();
    for (int i = 0; i < n; i++) {
        list.add(i);
    }
    for (int i = 0; i < list.size(); i++) {
        if (i % 2 == 0) {
            list.add(list.size() / 2, i); // insert (position, value)
        }
    }
    return list;
}
```

U. What is the order of the following method? _____

N = `list.size()`. The question uses the Java **LinkedList** class.

```
public static int u(LinkedList<Integer> list) {
    int result = 0;
    for (int i = 0; i < list.size(); i++) {
        for (int j = list.size() - 1; j >= 0; j--) {
            if (list.get(i) == list.get(j)) {
                result += list.get(i) - list.get(j);
            }
        }
    }
    return result;
}
```

V. The following values, from left to right, are inserted into an initially empty Red-Black tree using the algorithm demonstrated in class. Draw the resulting tree. Indicate if a node is red or black.

8 -2 3

W. In the small example used to illustrate the Huffman encoding algorithm, **"Eerie eyes seen near lake."** all the resulting codes had length 5 bits or less whereas all of the original codes for the characters were 8 bits in length.

Which of the following best explains why ALL the new codes are less than 8 bits in length? _____

A. Due to the inclusion of the pseudo end of file value.
B. Due to the small number of distinct characters in the text compared to the number of characters in ASCII.
C. Huffman coding always results in all codes being shorter than the original codes.
D. The frequency of e's and spaces were so much larger the frequencies of other characters
E. Due to using a fair priority queue to build the Huffman code tree instead of a heap based priority queue.

X. True or false, all problems have a dynamic programming solution? _____

Y. What is output by the following code? _____

```
int x = IntStream.of(1, 2, 5, 0, 1, -2, 3, -4)
        .map(n -> n * n)
        .filter(n -> n > 10)
        .map(n -> n / 2)
        .sum();
System.out.println(x);
```

Extra Credit. 1 point - Which programming assignment did you find most interesting?

_____

2. Graphs (18 points) - Complete a recursive helper method in the **Graph** class from assignment 11 that determines if it is possible to color the vertices of the **Graph** with 2 colors or not. (Note, this is one way of determining if a graph is bipartite or not.) Coloring the graph with 2 colors means each node is assigned **one of two colors** and adjacent vertices, those connected by a **single edge are always colored with different colors.**

For this question the **Graph** is unweighted and undirected. Every **Vertex** cost is set to 1.0 and if an **Edge** exists from **Vertex** A to **Vertex** B there shall be an **Edge** rom **Vertex** B back to **Vertex** A. Also, for this question assume the **Graph** is connected, every pair of vertices in the **Graph** has a path of one **or more** edges between them.

Use the **scratch** variable of the **Vertex** class to represent the color of a **Vertex**. 0 shall indicate the **Vertex** has not yet been colored.

The **Graph**, **Vertex**, and **Edge** classes for this question.

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

    // Sets scratch to 0 for all vertices
    private void clearAll(){ /* ... */ }


    public boolean canTwoColor() {
        clearAll();
        String start = vertices.keySet().iterator().next();
        // 1 represents the first color
        vertices.get(start).scratch = 1;
        return helper(start); // method to complete
    }


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

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

Complete the helper method called from the **`canTwoColor`** method.

The **`Graph`** object is not altered other than changing the **`scratch`** variable of Vertices.

You may use the provided **`Vertex`** and **`Edge`** classes. You may use methods on **`List`** objects including the for-each loop. Do not add any methods or fields to the **`Vertex`** or **`Edge`** classes.

The only method you may use from the Map interface in the recursive helper method is the **`Vertex get(String key`**) method on the **`vertices`** instance variable.

Do not add any new helper methods.

Do not create any new data structures or objects.

Do not use any other Java or classes.

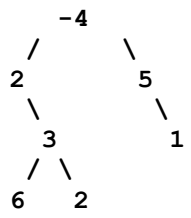
3 points. Explain your approach for the helper method:

Complete the helper method on the next page.

```
/* pre: currentName != null, currentName is the name of a Vertex in
        this Graph.
   post: per the problem description. */
private boolean helper(String currentName) {
```

3. Trees (14 points) - Complete an instance method named **`depthSum`** for an **`IntTree`** class that returns the sum of the values in nodes times the depth of that node. The **`IntTree`** class is a binary tree. The nodes of the tree store primitive **`ints`**.

If the calling object was the following tree:



then the method would return 39 = (-4 * 0) + (2 * 1) + (5 * 1) + (3 * 2) + (1 * 2) + (6 * 3) + (2 * 3)

Here are the **`IntTree`** and nested **`IntNode`** classes for this question:

```
public class IntTree {
    // only instance variable
    // root == null iff the tree is empty
    private IntNode root;

    private static class IntNode {
        private int val;
        // left and / or right store null if no child on that side.
        private IntNode left;
        private IntNode right;
```

Complete the following instance method for the **`IntTree`** class.

```
// pre: none, post: per the problem description.
// The calling object is not altered as a result of this call.
public int depthSum() {
```

Do no use any other Java classes or method in your answer. This includes native arrays.

You may not create or use native arrays of any length, not even 1.

Do not add any fields to the `IntTree` class.

Do not add any fields or methods to the `IntNode` class.

Complete the method on the next page.

```
// pre: none, post: per the problem description.
// The calling object is not altered as a result of this call.
public int depthSum() {
```

4. **Huffman Coding (18 points) -** Complete the following **constructor** for a **`HuffmanCodeTree`**. The class would be used to decode a file compressed with the Huffman encoding protocol used on assignment 10.

```
/*: pre: codes != null, codes[0].length = 2,
There are two columns in each row of codes.
None of the elements in codes are null.
All characters in the elements of the first column, the values, are between '0' and
'9' inclusive. All characters in the elements for the second column, the codes, are
'0' or '1'.
post: create a HuffmanCodeTree object based on the parameter codes. */
public HuffmanCodeTree(String[][] codes)
```

value	code
"46"	"11110"
"97"	"0101"
"115"	"1110"
"101"	"10"
"256"	"0100"

On the assignment you built the **`HuffmanCodeTree`** in one of two ways: using the frequency of values or using a binary representation of the tree. This question involves a new way of constructing the tree.

Each row in the parameter **`codes`** represents the **value** in a leaf node in column 0 and the **code** to get to that leaf node in column 1. A small (incomplete) example of the 2d array **`codes`** is shown to the right:

The structure of the tree is based on the given codes in the same way as presented in class and implemented on assignment 10, Huffman Coding.

The **`HuffmanCodeTree`** and **`TreeNode`** classes for this question:

```
public class HuffmanCodeTree {
    private TreeNode root; // Refers to root of the HuffmanCodeTree.
    private int numLeaves; // Number of leaves in this HuffmanCodeTree.

    private static class TreeNode {

        private int value;
        private int freq;
        private TreeNode left;
        private TreeNode right;

        public TreeNode(int val, int freq) {
            value = val;
            this.freq = freq;
        }
        // No other constructors or methods for this question.
    }
```

The **`freq`** field of all nodes shall be set to -1. The **`value`** field of internal nodes shall be set to -1. The **`value`** field of leaf nodes shall be set based on the values in the parameter **`codes`**.

You may use the charAt and length method for Strings and the static int Integer.parseInt(String s) method. You may, of course, use the length field for the given array.

You may use the provided `TreeNode` and `HuffmanCodeTree` class. Do not add any methods or constructors to the nested `TreeNode` class.

Do not use any other Java classes or methods.

Do not create any new data structures other than `TreeNodes`.

Do not use recursion in your answer.

```
public HuffmanCodeTree(String[][] codes) {
```