Name: _____ NetID: _____

- **WRITE your name and NetID on EVERY page.**
- **DO NOT REMOVE** the staple/crimp in your exam.
- **DO NOT BEGIN** until instructed to do so.
- WRITE NEATLY AND CLEARLY. If we cannot read your handwriting, you will not receive credit. Please plan your space usage. No additional paper will be given.
- This exam is worth 150 points.

Problem 1 – Miscellaneous (40 points)

1. **(10 points)** For this problem write the fastest algorithm (measured by worst-case big O). Describe an algorithm (give concise steps, do not write code) that given an unsorted array of length *m* and a sorted array of length *n*, finds the common items. Assume there are no duplicates in either array and, *m* is much larger than *n*. What is the worst-case big O running time?

2. **(6 points)** In the worst case, how many compares, in big O notation, to insert in a BST? Explain.

3. **(6 points)** In the worst case, how many compares, in big O notation, to insert in a d-way heap? Explain.

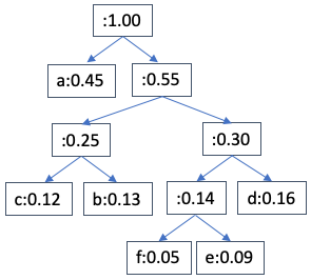
4. **(18 points)** On the Huffman Coding assignment the `TreeNode` class houses a `CharFreq` object "data" representing a certain character and its frequency, and `TreeNodes` "left" and "right" representing the left and right subtrees of the binary tree.

Name: _____ NetID: _____

```java
public class TreeNode {
    private CharFreq data;
    private TreeNode left;
    private TreeNode right;

    // We can create with data and both children
    public TreeNode(CharFreq d, TreeNode l, TreeNode r) {
        data = d;
        left = l;
        right = r;
    }

    // We can create with only data, children are null
    public TreeNode(CharFreq d) { this(d, l: null, r: null); }

    // No arguments sets everything to null
    public TreeNode() { this(d: null, l: null, r: null); }

    // Getters and setters
    public CharFreq getData() { return data; }
    public TreeNode getLeft() { return left; }
    public TreeNode getRight() { return right; }

    public void setData(CharFreq d) { data = d; }
    public void setLeft(TreeNode l) { left = l; }
    public void setRight(TreeNode r) { right = r; }
}
```

```java
public class CharFreq implements Comparable<CharFreq> {
    private Character character;
    private double probOcc;

    // We can set both the Character and double at once
    public CharFreq(Character c, double p) {
        character = c;
        probOcc = p;
    }

    // No arguments makes a null character and prob 0
    public CharFreq() { this(c: null, p: 0); }

    // Allows us to use Collections.sort() to sort by probOcc
    public int compareTo(CharFreq cf) {
        Double d1 = probOcc, d2 = cf.probOcc;
        if (d1.compareTo(d2) != 0) return d1.compareTo(d2);
        return character.compareTo(cf.character);
    }

    // Getters and setters
    public Character getCharacter() { return character; }
    public double getProbOcc() { return probOcc; }

    public void setCharacter(Character c) { character = c; }
    public void setProbOcc(double p) { probOcc = p; }
}
```

Write the following **RECURSIVE** `print` method that prints in preorder the probabilities in the Huffman Tree.



The `print` method prints the probabilities in this tree as follows:
1.0 0.45 0.55 0.25 0.12 0.13 0.30 0.14 0.05 0.09 0.16

```java
// Prints the probabilities according to a preorder
// traversal of the tree
private static void print (TreeNode root) {
      // COMPLETE THIS RECURSIVE METHOD
```

Name: _____ NetID: _____

Problem 2 - Priority Queue (16 points)

Suppose that a MAX heap is implemented using an array. The heap is used to keep track of the frequency of the words in a book, the word that appears most often in the book has the highest frequency and is located at the root (array index 1).

During input every word is inserted into the heap as follows: a word is read from the book and then the word is searched in the heap:

(i) if the word is not found it is then inserted into the heap with frequency 1.
(ii) if the word is found its frequency is increased by 1.

Answer the following questions.

a. **(4 points)** What is the worst-case running time, in big O notation, of searching for a word in the heap? Explain.

b. **(6 points)** Assume that a word has been searched and determined that it is not in the heap. What is the worst-case running time, in big O notation, of inserting a <u>new word</u> into the heap (after heap invariants are restored)? Explain.

c. **(6 points)** Assume that a word has been searched and determined that it is in the heap. What is the worst-case running time, in big O notation, to update a word's frequency (heap invariants are expected to be restored after insert is complete)? Explain.

Name: _____ NetID: _____

Problem 3 - Hash table (20 points)

The following keys will be inserted in sequence to a hash table. For simplicity, we omit the "values" associated with the keys.

14 8 27 10 15 90 11 7 12 17

1. **(16 points)** Assume the Separate-Chaining Symbol Table API discussed in class is used. The table size is denoted by `m` and the hash function is `hash(key) = key % m`. The initial table size is 3. *Note that keys are inserted at the front of the list*. The threshold of the load factor is 2. So, when the load factor is larger than 2, rehashing should be performed. Suppose we would double the table size when we do rehashing. Show the contents of the two hash tables before rehashing and after rehashing.

2. **(4 points)** What is the running time (big O) for rehashing given the input size n? Give the reasoning.

Name: _____ NetID: _____

Problem 4 – Graph (40 points)

1. **(6 points)** Assume an **undirected** graph G with 7 vertices and 8 edges (*v*,*w*) (an edge between vertices *v* and *w*). Based on the adjacency list Java implementation discussed in class where a *new edge is added to front of list*. If the list of 8 edges (0, 1) (1, 2) (2, 4) (2, 3) (3, 6) (5, 0) (3, 1) (3, 4) were added in sequence to construct the graph G, show the vertex-indexed array of lists.

2. Answer the following questions based on the adjacency list in problem 4.1.

   a. (2 points) If a single for loop is used to iterate over the list of vertices adjacent to a given vertex *v*, what is the number of iterations?

   b. (3 points) Given the code below, what is the number of times `StdOut.println` statement is executed, assuming G is the **undirected graph** constructed in problem 4.1?

```
for ( int v = 0; v < G.V(); v++ )
    for ( int w : G.adj(v) )
         StdOut.println( v + "-" + w );
```

Name: _____ NetID: _____

3. Answer the following questions based on the Java code below and the adjacency list representation of graph G constructed in problem 4.1.

```
private void dfs(Graph G, int v) {
    marked[v] = true;
    for (int w : G.adj(v)) {
        if (!marked[w]) {
            edgeTo[w] = v;
            dfs(G, w);
        }
    }
}
```

a. (5 points) Write the sequence of vertices visited of a method call to `dfs(G, 0)`.

b. (5 points) write the contents of the array `edgeTo[]` as a consequence of the method call in 3.a.

Name: _____ NetID: _____

4. Answer the following questions based on the Java code below and the adjacency list representation of graph G constructed in problem 4.1.

```
private void bfs(Graph G, int s) {
        Queue<Integer> q = new Queue<Integer>();
        distTo[s] = 0;
        marked[s] = true;
        q.enqueue(s);

        while (!q.isEmpty()) {
            int v = q.dequeue();
            for (int w : G.adj(v)) {
                if (!marked[w]) {
                    edgeTo[w] = v;
                    distTo[w] = distTo[v] + 1;
                    marked[w] = true;
                    q.enqueue(w);
                }
            }
        }
    }
```

a. (5 points) Write the sequence of vertices visited of a method call to `bfs(G, 0)`.

b. (5 points) Write the contents of the array edgeTo[] as a consequence of the method call in 4.1.

Name: _____ NetID: _____

5. Assume a **directed** graph G with 7 vertices and 8 edges (*v*,*w*) (an edge between vertices *v* and *w*). Based on the adjacency list Java implementation discussed in class where a *new edge is added to front of list*.

   a. (6 points) If a list of 8 edges (0, 1) (1, 2) (2, 4) (2, 3) (3, 6) (5, 0) (3, 1) (3, 4) with 7 vertices were added in sequence to construct the graph G, show the vertex-indexed array of list.

   b. (3 points) What is the maximum number of edges one can add to the directed graph in 5.1?

Name: _____ NetID: _____

Problem 5 - Sorts (34 points)

1. **(5 points)** Which sorting algorithm would you use, insertion sort, selection sort, merge sort or quick sort, to sort a large array that is known to be almost sorted? Justify your answer.

2. **(2 points)** Which sorting algorithm sorts an array by cutting the array in half, recursively sorting each half, and then merging the sorted halves?

3. **(3 points)** Explain how does merge sort compare to heap sort, in big O notation, with respect of storage consumption?

Name: _____ NetID: _____

4. **(24 points)** Trace the quicksort algorithm on the following array. Use the first item as the pivot when doing a split.

   25, 8, 89, 28, 15, 9, 2

   a. **(12 points)** Show the series of item swaps that are performed in the FIRST split process, and the array after each of these swaps, up to and including the step of placing the pivot at its correct location. You only need to show the FIRST split of the original array.

   b. **(12 points)** Show the full recursion tree, i.e. all splits, on original array and all subarrays.