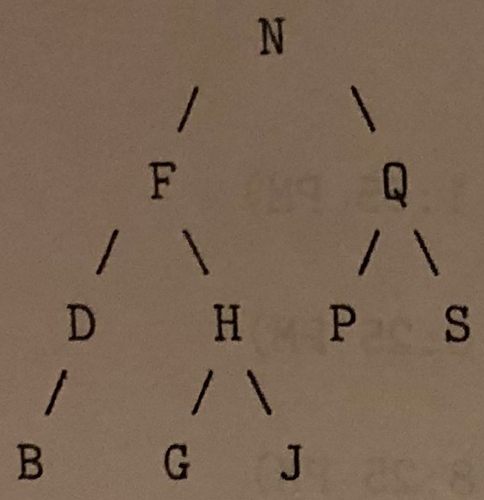
1. Trees/Lists (30 pts;8+9+8+5)

a) Mark balance factors in the nodes of the following AVL tree in which letters are arranged in alphabetical order. Then insert M, and show all the steps of how you apply the AVL insertion algorithm to arrive at the final AVL tree.



b) Complete the following *recursive* method to tell if two binary trees are isomorphic (have the same shape, data is irrelevant). You may NOT use helper methods.

```
public class BTNode { int data; BTNode left, right; ... }
public static boolean isomorphic(BTNode root1, BTNode root2) {
   // COMPLETE THIS RECURSIVE METHOD
```

CS 112 Spring 2018 Final Exam; netid: _____

c) Two playlists of songs are stored in UNSORTED doubly-linked lists of lengths m and n, respectively. What is the best case big O running time to print the common songs in the playlists? Describe your algorithm (list the steps), explain the best case scenario, and derive the running time (make sure to list the unit-time operations you are counting). You are allowed to modify either list.

d) A Huffman tree is built for 5 characters. Come up with probabilities for these characters that results in the tallest possible Hufffman tree. Show the probabilities as well as the Huffman tree.

3. Heap/Sorting (30 pts;6+10+8+6)

a) Complete the heapify (build heap) method in the following MaxHeap class.

```
public class MaxHeap {

    // sifts down in an array starting at index k, ASSUME IMPLEMENTED
    private static void siftDown(int[] list, int k) { ... }

    // converts an array into a heap with repeated sift downs
    public static void heapify(int[] list)  {  // COMPLETE THIS METHOD
```

b) Suppose that we have k heaps, with n elements in each. We wish to combine these into a single heap. We pair them into $k/2$ pairs, and apply the fastest technique to combine each pair, resulting in $k/2$ heaps. We then repeat this process by pairing into $k/4$ pairs and combining each pair with the fastest possible algorithm, and so on, until we are left with a single heap. What is the worst-case big O running time? Derive with detailed reasoning.

c) Trace the quicksort algorithm on the following array, and show the recursion tree. Use the first item as the pivot when doing a split.

24 6 72 28 12 9 3

d) In quicksort, after every split, always recurse on the smaller sub array first. Why?

c) From the AVL tree created above, you want to print all words, with counts, grouped by word length. (For instance, `is:2, the:2, and:1, this:2, line:3`). In any group of words of the same length, the printout can be in any order (e.g. *and* does not have to come before *the*.)

Describe the fastest algorithm (with whatever supporting data structures you need), to print the output. Derive (show your work!) the worst case time big O running time of your algorithm, assuming the maximum word length is m.