**Unit #3: Data Structures** **Accuracy**
**Lab 3.2: Huffman Encoding**

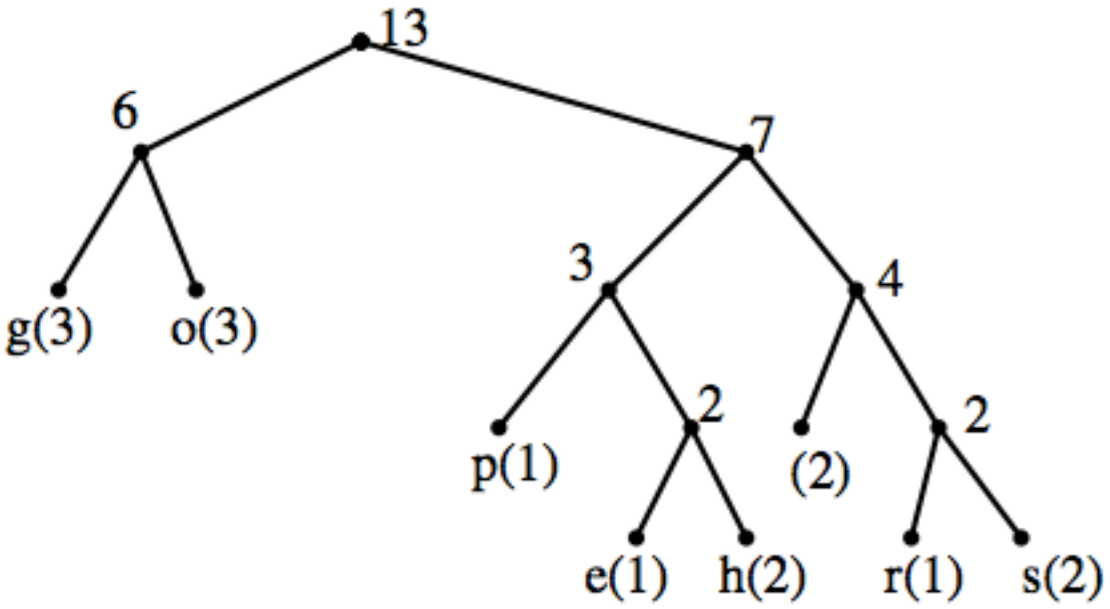
**Background**

Huffman encoding is a technique for compressing files. The idea is to build a binary tree whose leaves represent the unique characters of the file and whose branches represent the bits of the code.

For example, imagine a file contains: go go gophers which is a 13-character, 104 bit sequence with the character distribution (g:3, o:3, " ": 2, h:1, e:1, p:1, r:1, s:1).

A Huffman tree is generated by building a leaf for each unique character, then repeatedly adding nodes together with the smallest counts, until all of the nodes are in a tree stored by one node.

This process starts with a priority queue (heap) of Huffman Nodes and progresses to a tree like the one below from our example.




With this tree the original file can be encoded into 37 bits, as follows:
g(00)o(01) (110)g(00)o(01) (110)g(00)o(01)p(100)h(1011)e(1010)r(1110)s(1111)
or 0001110000111000011001011101011101111.

The steps taken to get from the root of the tree to the leaf give the binary representation of each letter. Following down a left path yields a 0 and following down a right path yields a 1.

**Encoding:**

1. Get character count and create Huffman Nodes out of them all and put them all in a heap with nodes with smallest count given highest priority.
2. While there are still multiple nodes on the heap, add the next smallest nodes together creating a new node that is the sum of the counts. This node can have the character '*' and will have the two "added" nodes as left and right children. With the highest priority on the left.
3. Once this is finished there will be one node left that is actually a HuffmanTree (though it is of type HuffmanNode).
4. The encoding of each letter is the "directions" to get to that letter from the root of the tree. That is, if you starting at the root you "take" two lefts then a right to get to the letter, it's encoding is 001. I recommend you write a function that takes a letter and the heap and gets the encoding.
5. Encode each letter in the message and create a single encoded string.
6. Now you are ready to output the encoded message and the tree that goes with it (so someone reading your message can decode it). You should output the tree as a String of characters that is the preorder traversal of the tree.

**Decoding:**

1. First you must take in the encoded message and the string representation of the tree.
2. Create a Huffman tree from the string representation.
3. Start at the root of the tree and begin following the "directions" of the encoded message until you find your first letter. Then, cut the directions you have followed off the encoded message and add the letter you found to an array of chars. Go back to the root of the tree and start again.
4. Finally you will have a char array of your message. Convert that to a string and output.