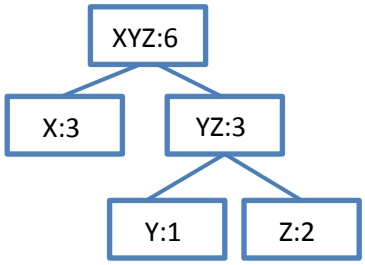
605.202 Data Structures LAB 3

Using the frequency table shown below, build a Huffman Encoding Tree. Resolve ties by giving single letter groups precedence (put to the left) over multiple letter groups, then alphabetically. Do not worry about punctuation or capitalization.

For this lab only, you may use string methods to work with the groups of letters.

This program will need to be able to read three different files: a file containing the frequency table, a file containing clear text to be encoded and a file containing encoded strings.

Print out the tree by doing a preorder traversal. Print the resulting encoding generated by the tree. An example is given below for a 3-letter alphabet. You may use any reasonable format to print the encoding.



X – 3
Y – 1
Z – 2

The tree in preorder is: XYZ: 6, X: 3, YZ: 3, Y: 1, Z: 2

The code is X = 0; Y = 10, Z = 11;

In your write-up, consider whether you achieved any useful data compression with this method. Compare to conventional encoding. How would your results be affected by using a different scheme to break ties, for example, if you had given precedence to alphabetical ordering and then to the number of letters in the key? What other structures did you use and why?

Encode the strings in the supplied ClearText.txt file, plus several others of your choice:

Decode the strings the supplied Encoded.txt file, plus some others of your choice:

Use the frequency table in the supplied FreqTable.txt file. Consider experimenting with other frequency tables.

As a check, here is a simple string in clear text and encoded form.

Hello World 1101101000010001111100011111101000000101100