Lab6 Huffman Algorithm Implementation Ideas.

You read text from file, for each character you count frequency, then you create Huffman tree based on this frequency
Then you find code for each character based on the path to leaf for each character, then you encode your original file and then decode from encoded file and get encoded file that must be the same as original file.
Some things that you might want to know:
1.When you implement Huffman tree:
You might want to order by frequency and create a list with nodes with characters and corresponding frequencies, then take two nodes with smallest (frequency) nodes and connect them to each other (like a fork), create node as root and remove these two nodes from the list and insert the node – root into the list and as frequency(weight) in the place in the list, so list stays ordered. You can do it recursively until you have just one node in the list – it will be root in the Hoffman tree (tree is built)
2. you can write code for all characters using Huffman tree.(array or vector) Now you can encode the original file.
3 to decode you can use Huffman tree.
4. Can be situation them weight of nodes are the same. In this situation you can sort using characters also, because all ASCII codes are different, but it is not convenient.
You can save table with your encoding in the beginning of your encoded file, (length of the table, also, save before the table), so you can read the table (since you know the length of the table), read char, encoding sequence, you can create Huffman tree for decoding, and then, based on this tree, decode this encoded text.

