CS222
Assignment - 6

message.txt file is opened in the main function and the string to be coded is read.

generateFreq function is called to count the occurrences of each character in the message using a map named freq.(contains key as a character and value as a pair of frequency of each character and binary coded string initiated to empty string).

encodeData message is called to create the huffman Table and encode the message and write it into encoded_data.txt

A priority queue pq(min heap) is initiated.All the huffman nodes(consists of character, frequencies, left and right huffman nodes) are pushed into pq.
Size of the priority queue is written into the file.(number of different characters in the message).

generateTree function is called to generate the huffman tree which is used to code and decode the message.(the root node of this tree is kept as a global variable).
The top two minimum frequency valued nodes are popped and a new node with frequency as sum of their frequencies and left and right nodes as the popped nodes is added into the pq.
This process is repeated until one node is left in the pq and that is the root node of the huffman tree.

generateCodes function is called to generate the binary code for each character and store it in the map.
This is done using recursion, if the node has left child, it is considered as 0 and moves to the left, if the node has a right child, it is considered as 1 and moves to the right.
If the node is the leaf node, the generated string so far is the binary code of the character in the leaf node and is written into the huffman table.
Encoded_data.txt is opened and the binary string of each character is fetched from the freq map and is written into the file.

To decode the string, decodeData function is called, which uses the huffman tree and the encodedMessage.
For each character in the encodedMessage, if the character is ‘0’, it moves to left and if the character is ‘1’, it moves to the right and when a leaf node is reached, the character in that leaf node is added to the decodedString. Starts from the root node after every character addition to the decodedString.

TimeComplexity - O(nlogn) - encoding, (nlogn) - decoding. (n - number of unique characters in the message).