This is an individual CA. It is due on 26 April. You may be asked to demo and explain your solutions. If you use code from any source, you must comment and cite it correctly, otherwise you may get zero marks. *Ignorantia juris non excusat.* Here are examples of how to cite other people's code
https://elearning-ta.tudublin.ie/mod/page/view.php?id=242177
and http://uark.libguides.com/content.php?pid=155080&sid=1780817

Do not use Structs in your code. Do not use a min or max heap class. You should only have Huffman Tree and Huffman Tree Node classes, and you are expected to use the relevant template classes in the STL. For example, the STL Pair, Map and Priority Queue containers.

You are required to carry out the tasks below to develop an optimal **Huffman Tree** with which to compress a file of ASCII text.

You should aim to complete the first section of this problem (i.e., the steps 1 through 5 below). Once you have completed tasks 1-5, then complete 6 & 7, and finally 8.

Section A (40 Marks)

The first 5 tasks will encode and decode the file into a string of 0's and 1's using a Huffman tree.

1. Given a text file, determine the frequency of each character in the text (**map** of character and frequency).
2. Build an optimal Huffman tree to represent these characters with these frequencies (maintain a **priority queue of trees**, removing and joining trees until only one tree remains – the final Huffman tree required) Hint: each Hoffman Tree should have a weight data member, as well as a pointer to its root node.
3. Use the tree to map each character to the string of 0's and 1's needed to encode it (pre-order traversal of tree, store results in a **map** of character and string).
4. Use the map to encode the text to a string of 0's and 1's and write to an encoded file
5. Use the Huffman Tree to decode the text and write to another file.

Section B (20 Marks)

Tasks 6 and 7 compress and un-compress the file:

6. To compress: break up the string of 0's and 1's into 8-bit chunks, and write the character represented by each chunk to the compressed file.
7. Un-compress by replacing each character with the 8 bits needed to represent it.

Section C (30 marks)

Task 8:

8. Embed the Huffman Tree used for the encryption as the first bytes of the compressed file and retrieve it as part of the decompression process.

10 marks for good programming practice.

Marks may be deducted for bad programming practice [-80 Marks]

For example, mixing implementation and declaration, including .cpp, files, global variables, etc, etc