**Project 3**

In this project I learned a lot how min heaps can work with other methods to build something, in this case compressing a file. I also learned how to traverse, create, and utilize a Huffman tree for compression. This project also helped me become more familiar with hash tables and using them with keys.

Data Structures used:

I used a hashtable to store the code associated with each ascii which made looking up each each char value faster. I used my own min heap of type hufftree to prioritize the root weight in order to build the hufftree. The hufftree class consisted of a root with a weight value. The root was of type huffnode, which contained a weight, a left and right node, and a value. If the left and right were null then it contained an ascii value in value, otherwise its value would be null. The frequency table was stored in an array where an ascii value’s frequency was directly indexed to its bin.

Computational complexities:

encodeFile – O(n) where n is number of chars in text file (hashtable is used to lookup each code)

decodeFile – O(nlogk) where n is number of 0’s and 1’s to decode and k is number of nodes in hufftree

traverseHuffman – O(n) where n is number of ascii values (hashtable is used to print each char and its code