CSE4014 – Data Structures & Algorithms Project (A)
Huffman Coding

```
#include <iostream> : This library for the standart i/o functions
#include <fstream> : For writing/reading functions
#include <map> : For using map data structure
#include <string> : For using string functions
#include <queue> : For using queue data structure
#include <bitset> : For dec to binary converting func
#include "HuffManHeap.h" : calling the header file
```

This project reads the spesific named text as “text.txt” ,and maps all chars into a map type convert them into a priority queue with HuffmanNode type which is defined in `HuffManHeap.h` And creating a huffman tree after that with using that tree finding and storing the codes for each unique char into a map.Finally Writing the huffman codes and ascii codes into seperate text files named “huffman.txt” and “ascii.txt” and printing them in screen.

Definitions of all functions

```
>>>map<char, int> readThefile()
```

This function created for reads the all chars in the text file.The if state was for controlling the file is exist or not. The “noskipws” in while loop for taking spaces in the text.In the loop “`freq[myWord]++`” will create a key-value relation for every char and it will be upgraded according their frequencies.
The function is map type function so it means it returns the map typed variable which name is freq it will be use in the main.

Space Complexity is O(n),there is just one whilee loop fills the map 2n -> O(n).
Also Time complexity is O(n) since there is just one straiht loop without nested.

```
>>>void create_HuffMan_tree(priority_queue<HuffmanNode*, vector<HuffmanNode*>, myComparator>
&HuffmanTree)
```

This function has 1 parameter named huffmantree.it is priority queue typed variablew which includes huffmannode typed pointer for taking root node and vecttor for storing other ndoes.Also it has a comparator for ordering with special purpose of huffman algorithm. The while loop will loops till the huffmanTree size will 1,that means the only node is the root node so the huffman tree have been created.

Has 3 local variables for keeping the nodes while traversing. Since the only way is poping for traversing in priority queue.First takinf the top for left child and pop it after that the priority queue

will be re ordered and taking the new top for right child. So for creating a huffman node we copying the left and right child the topNode variable and their frequency is setting with the sum of the right and left nodes frequency so its a huffman node and we are pushing it the huffmantree.

Time complexity : O(nlogn) The while loop loops linearly it causes the O(n) also the priority queue has comparator algorithm and it causes the O(logn).
So finaly time complexity is O(nlogn)

>>>void calculateTheCodes(class HuffmanNode* rootNode, string str, map<char, string> &codes)

This function created calculate the huffman codes via huffmantree.
It has 2 parameters 1 for rootnode for accesing huffman tree.Other one is a map reference for taking by main function to store the all codes in a map variable.
It is a recursive function if the root node will be null it returns .If rootNode->data = 0 that means the null for a char typed variable so if its not null that means it is a leaf and that means we have calculated the actual huffman value.

Time Complexity: O(logn) Since this function is using recursively it causes a loop so it has a time complexity.It has called 2 times recursively one for left ,another one is for right node.So that means We are not traversing lineary in heap.In every step we are checking more nodes that means its logarithmic ,if its not a leaf node whic is controlling via if state "if (rootNode->getData() != 0" (0 is a special indicator here if its not null the loop can realise it is a leaf so it reached the actual huffman code) the function keeps to dive to deeps till find the huffmancode via branches (left means add 0 ,right means add 1).

>>>>void printTextAs_HuffManTree(map<char, string>codes)

This function has one parameter codes named it is the map for the all huffmantree codes which stored in.
Firstly function will open the sample file and creates a file named "huffman.txt".
The function printing the codes on screen and write them in a file at same time in while loop.

Time Complexity:O(n) The while loop loops linearly.

>>>>>void printTextAs_Ascii()

The function ahs same logic with " void printTextAs_HuffManTree"

the only difference is here there is int type casting for char variable in while loop we take the decimal ascii value here and we call the dec to binary func to get the actual binary value .

Time Complexity:O(n) The while loop loops linearly.and it calls a func from bitset library which has O(1) time complexity so its not effect the total complexity.

>>>>`void printTheCodes(map<char, string>& codes)`

This function traverse over the map and prints thie actual huffman codes.

The time Complexity is O(n) cause of the linear for loop

FINAL TIME COMPLEXITY

The final time complexity is O(nlogn),

Assume there are n unique chars in text file, `create_HuffMan_tree` func will take O(n) also inside the loop the priority queue comparater will re order every element for every step and it takes O(logn) so sum is O(nlogn) ,This func dominates all others so the final complexity is O(nlogn)