### Design and Analysis of Algorithms Assignment - 8

###### Name: Dhanraj Kore

###### Div: TY B

###### Roll No: 60

###### Batch : B-3

**Huffman Encoding**

**CODE :**

#include<bits/stdc++.h>

using namespace std;

struct Node

{

    char data;

    int freq;

    Node\* left;

    Node\* right;

};

Node\* createNode(char d,int f)

{

    Node\* temp = new Node;

    temp->data = d;

    temp->freq = f;

    temp->left = NULL;

    temp->right = NULL;

    return temp;

}

void show(Node\* root)

{

    if(root)

    {

        show(root->left);

        cout<<root->freq<<" ";

        show(root->right);

    }

}

void generateCodes(Node\* root,string s)

{

    if(!root)

        return;

    if(root->data!='$')

        cout<<"\n\t\t"<<root->data<<" "<<s<<endl;

    generateCodes(root->left,s+'0');

    generateCodes(root->right,s+'1');

}

void hauffmanTree(vector<pair<char,int>> &v)

{

    priority\_queue<Node\*,vector<Node\*>,greater<Node\*>> p;

    for(int i=0;i<v.size();i++)

    {

        Node\* temp = new Node;

        temp = createNode(v[i].first,v[i].second);

        p.push(temp);

    }

    Node \*left,\*right,\*top;

    while(p.size()!=1)

    {

        left = p.top();

        p.pop();

        right = p.top();

        p.pop();

        top = new Node;

        top = createNode('$',left->freq+right->freq);

        top->left = left;

        top->right = right;

        p.push(top);

    }

    cout<<"\n\tInorder traversal of Hauffman Tree is: ";

    show(p.top());

    cout<<endl<<"\n\tCodes of each character : "<<endl;

    generateCodes(p.top(),"\t");

    cout<<endl;

}

bool helper(pair<char,int> &p1,pair<char,int> &p2)

{

    return p1.second<p2.second;

}

int main()

{

    string s;

    cout<<"\n\tEnter the input string : ";

    cin>>s;

    unordered\_map<char,int> m;

    for(int i=0;i<s.size();i++)

        m[s[i]]++;

    vector<pair<char,int>> v;

    for(auto i:m)

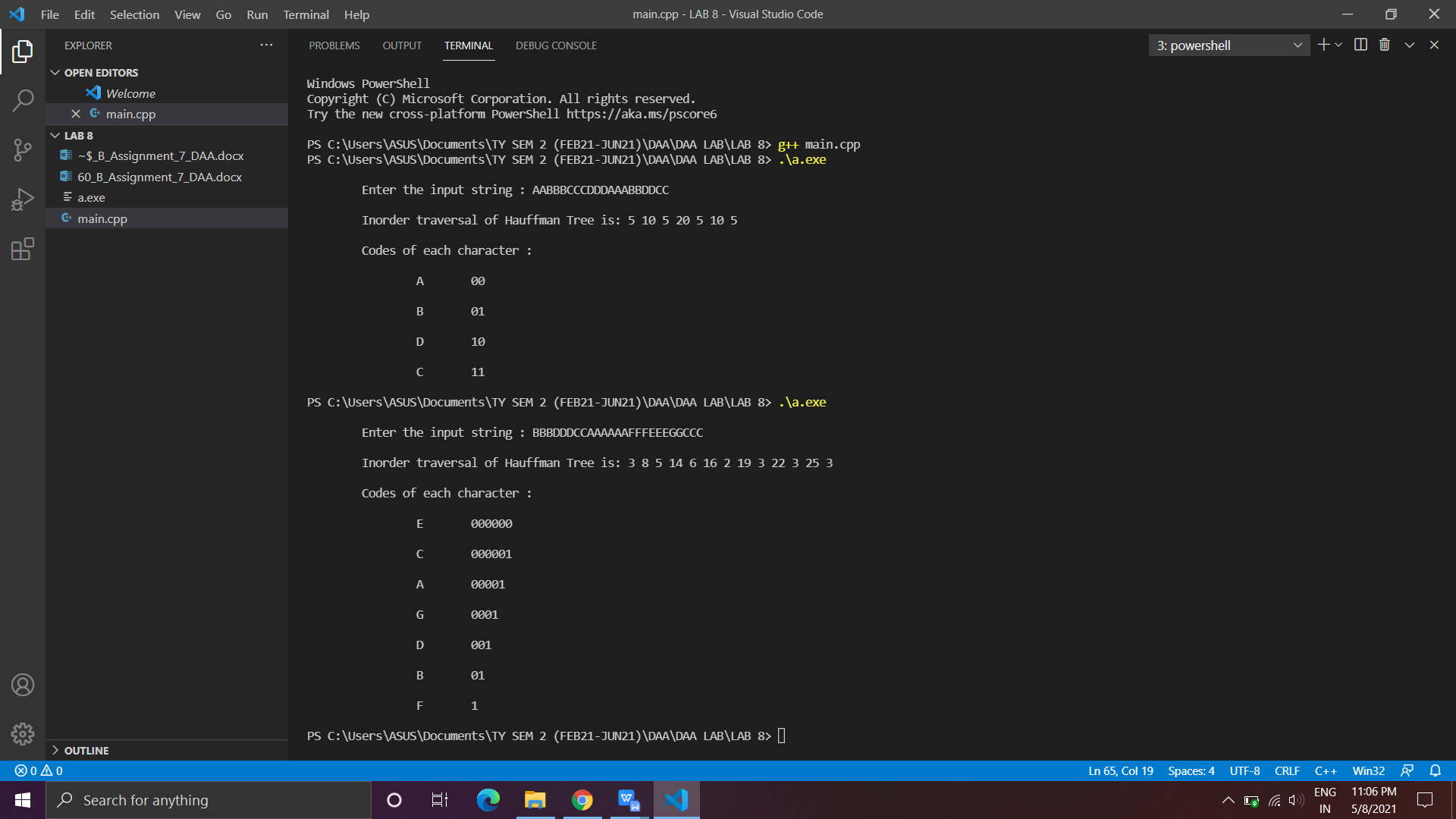
        v.push\_back(i);

    sort(v.begin(),v.end(),helper);

    hauffmanTree(v);

}

**O/P:**



**Time Complexity:** **O(nlogn)**

**Space Complexity: O(n)**