**ASSIGNMENT-7**

**EXPERIMENT – 12: Write a program to implement Huffman’s coding using greedy approach.**

#include <iostream>

#include <vector>

#include <queue>

#include <string>

using namespace std;

class Huffman\_Codes

{

struct New\_Node

{

char data;

size\_t freq;

New\_Node\* left;

New\_Node\* right;

New\_Node(char data, size\_t freq) : data(data),

freq(freq),

left(NULL),

right(NULL)

{}

~New\_Node()

{

delete left;

delete right;

}

};

struct compare

{

bool operator()(New\_Node\* l, New\_Node\* r)

{

return (l->freq > r->freq);

}

};

New\_Node\* top;

void print\_Code(New\_Node\* root, string str)

{

if(root == NULL)

return;

if(root->data == '$')

{

print\_Code(root->left, str + "0");

print\_Code(root->right, str + "1");

}

if(root->data != '$')

{

cout << root->data <<" : " << str << "\n";

print\_Code(root->left, str + "0");

print\_Code(root->right, str + "1");

}

}

public:

Huffman\_Codes() {};

~Huffman\_Codes()

{

delete top;

}

void Generate\_Huffman\_tree(vector<char>& data, vector<size\_t>& freq, size\_t size)

{

New\_Node\* left;

New\_Node\* right;

priority\_queue<New\_Node\*, vector<New\_Node\*>, compare > minHeap;

for(size\_t i = 0; i < size; ++i)

{

minHeap.push(new New\_Node(data[i], freq[i]));

}

while(minHeap.size() != 1)

{

left = minHeap.top();

minHeap.pop();

right = minHeap.top();

minHeap.pop();

top = new New\_Node('$', left->freq + right->freq);

top->left = left;

top->right = right;

minHeap.push(top);

}

print\_Code(minHeap.top(), "");

}

};

int main()

{

int n, f;

char ch;

Huffman\_Codes set1;

vector<char> data;

vector<size\_t> freq;

cout<<"Enter the number of elements:";

cin>>n;

cout<<"Enter the characters:-";

for (int i=0;i<n;i++)

{

cin>>ch;

data.insert(data.end(), ch);

}

cout<<"Enter the frequencies \n";

for (int i=0;i<n;i++)

{

cin>>f;

freq.insert(freq.end(), f);

}

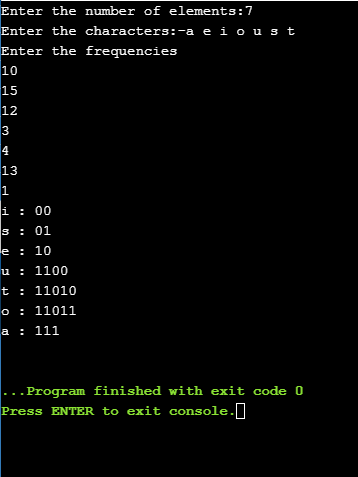
size\_t size = data.size();

set1.Generate\_Huffman\_tree(data, freq, size);

return 0;

}

**OUTPUT-**

****

**EXPERIMENT – 13: Write a program to find minimum Number of Platforms Required for a Railway/Bus Station**.

#include<iostream>

#include<algorithm>

using namespace std;

int findPlatform(int arr[], int dep[], int n)

{

sort(arr, arr+n);

sort(dep, dep+n);

int plat\_needed = 1, result = 1;

int i = 1, j = 0;

while (i < n && j < n)

{

if (arr[i] < dep[j])

{

plat\_needed++;

i++;

if (plat\_needed > result)

result = plat\_needed;

}

else

{

plat\_needed--;

j++;

}

}

return result;

}

int main()

{

int arr[30],dep[30],n,i;

cout<<"\nEnter the size of the array:";

cin>>n;

cout<<"\nEnter arrival time:-\n";

for(i=0;i<n;i++){

cin>>arr[i];

}

cout<<"\nEnter departure time:-\n";

for(i=0;i<n;i++){

cin>>dep[i];

}

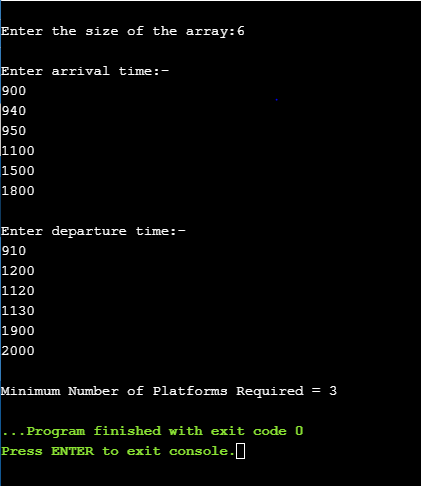
cout <<"\nMinimum Number of Platforms Required = "

<< findPlatform(arr, dep, n);

return 0;

}

**OUTPUT-**

****