**Practical No : 3**

**Name :** Samrudhi Pawar

**Roll No :** 44

**Title :** Develop a program to implement Huffman Encoding using a greedy strategy.

**Program :**

#include <bits/stdc++.h>

using namespace std;

// A Huffman tree node

struct MinHeapNode {

// One of the input characters

char data;

// Frequency of the character

unsigned freq;

// Left and right child

MinHeapNode \*left, \*right;

MinHeapNode(char data, unsigned freq)

{

left = right = NULL;

this->data = data;

this->freq = freq;

}

};

struct compare {

bool operator()(MinHeapNode\* l, MinHeapNode\* r)

{

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

}

};

// Prints huffman codes from

// the root of Huffman Tree.

void printCodes(struct MinHeapNode\* root, string str)

{

if (!root)

return;

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

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

printCodes(root->left, str + "0");

printCodes(root->right, str + "1");

}

// The main function that builds a Huffman Tree and

// print codes by traversing the built Huffman Tree

void HuffmanCodes(char data[], int freq[], int size)

{

struct MinHeapNode \*left, \*right, \*top;

// Create a min heap & inserts all characters of data[]

priority\_queue<MinHeapNode\*, vector<MinHeapNode\*>,

compare>

minHeap;

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

minHeap.push(new MinHeapNode(data[i], freq[i]));

// Iterate while size of heap doesn't become 1

while (minHeap.size() != 1) {

// Extract the two minimum

// freq items from min heap

left = minHeap.top();

minHeap.pop();

right = minHeap.top();

minHeap.pop();

top = new MinHeapNode('$',

left->freq + right->freq);

top->left = left;

top->right = right;

minHeap.push(top);

}

// Print Huffman codes using

// the Huffman tree built above

printCodes(minHeap.top(), "");

}

// Driver Code

int main()

{

char arr[] = { 'a', 'b', 'c', 'd', 'e', 'f' };

int freq[] = { 5, 9, 12, 13, 16, 45 };

int size = sizeof(arr) / sizeof(arr[0]);

HuffmanCodes(arr, freq, size);

return 0;

}

**Output :**

f: 0

c: 100

d: 101

a: 1100

b: 1101

e: 111