#include <iostream>

#include <queue>

#include <map>

using namespace std;

// Huffman tree node

struct Node {

char data;

int freq;

Node\* left;

Node\* right;

Node(char data, int freq) : data(data), freq(freq), left(nullptr), right(nullptr) {}

};

// Comparison function for priority queue

struct Compare {

bool operator()(Node\* a, Node\* b) {

return a->freq > b->freq;

}

};

// Build Huffman tree and return the root

Node\* buildHuffmanTree(string characters, int freq[], int n) {

// Priority queue to store the nodes of the Huffman tree

priority\_queue<Node\*, vector<Node\*>, Compare> pq;

// Create leaf nodes for each character and add them to the priority queue

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

Node\* newNode = new Node(characters[i], freq[i]);

pq.push(newNode);

}

// Build Huffman tree

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

// Extract two minimum frequency nodes

Node\* left = pq.top();

pq.pop();

Node\* right = pq.top();

pq.pop();

// Create a new internal node with sum of frequencies

Node\* internalNode = new Node('$', left->freq + right->freq);

internalNode->left = left;

internalNode->right = right;

// Add the new internal node back to the priority queue

pq.push(internalNode);

}

// The remaining node is the root of the Huffman tree

return pq.top();

}

// Print Huffman codes in preorder traversal of the tree

void printHuffmanCodes(Node\* root, string code) {

if (root == nullptr)

return;

// Leaf node encountered, print the character and its code

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

cout << " " << root->data << " : " << code << endl;

}

// Traverse left subtree with code '0'

printHuffmanCodes(root->left, code + "0");

// Traverse right subtree with code '1'

printHuffmanCodes(root->right, code + "1");

}

int main() {

string characters = "abcdef";

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

int n = sizeof(characters) / sizeof(characters[0]);

Node\* root = buildHuffmanTree(characters, freq, n);

cout << "Huffman codes will be:" << endl;

printHuffmanCodes(root, "");

return 0;

}