#include <iostream>

#include <queue>

#include <vector>

using namespace std;

// Define the structure for a node in the Huffman tree.

struct HuffmanNode {

char data;

int frequency;

HuffmanNode\* left;

HuffmanNode\* right;

HuffmanNode(char d, int f) : data(d), frequency(f), left(nullptr), right(nullptr) {}

};

// Define a custom comparator for the priority queue based on frequency.

struct CompareHuffmanNode {

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

return a->frequency > b->frequency;

}

};

// Function to print Huffman codes in preorder traversal.

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

if (root == nullptr) return;

if (root->data != '\0') {

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

}

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

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

}

int main() {

string S = "abcdef";

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

int N = S.length();

// Create a priority queue for Huffman nodes.

priority\_queue<HuffmanNode\*, vector<HuffmanNode\*>, CompareHuffmanNode> pq;

// Create nodes for characters and their frequencies and add them to the priority queue.

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

pq.push(new HuffmanNode(S[i], f[i]));

}

// Build the Huffman tree.

while (pq.size() > 1) {

HuffmanNode\* left = pq.top();

pq.pop();

HuffmanNode\* right = pq.top();

pq.pop();

HuffmanNode\* newNode = new HuffmanNode('\0', left->frequency + right->frequency);

newNode->left = left;

newNode->right = right;

pq.push(newNode);

}

HuffmanNode\* root = pq.top();

// Perform a preorder traversal to print Huffman codes.

string code = "";

printHuffmanCodes(root, code);