/\* HUFFMAN ENCODING \*/

#include <iostream>

#include <queue>

#include <unordered\_map>

using namespace std;

struct Node {

char ch;

int freq;

Node \*left, \*right;

};

Node\* getNode(char ch, int freq, Node\* left, Node\* right) {

Node\* node = new Node();

node->ch = ch;

node->freq = freq;

node->left = left;

node->right = right;

return node;

}

struct comp {

bool operator()(Node\* l, Node\* r) {

return l->freq > r->freq;

}

};

void encode(Node\* root, string str, unordered\_map<char, string> &huffmanCode) {

if (root == nullptr)

return;

if (!root->left && !root->right) {

huffmanCode[root->ch] = str;

}

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

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

}

void decode(Node\* root, int &index, string str) {

if (root == nullptr) {

return;

}

if (!root->left && !root->right) {

cout << root->ch;

return;

}

index++;

if (str[index] == '0')

decode(root->left, index, str);

else

decode(root->right, index, str);

}

void buildHuffmanTree(string text) {

unordered\_map<char, int> freq;

for (char ch: text) {

freq[ch]++;

}

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

for (auto pair: freq) {

pq.push(getNode(pair.first, pair.second, nullptr, nullptr));

}

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

Node \*left = pq.top(); pq.pop();

Node \*right = pq.top(); pq.pop();

int sum = left->freq + right->freq;

pq.push(getNode('\0', sum, left, right));

}

Node\* root = pq.top();

unordered\_map<char, string> huffmanCode;

encode(root, "", huffmanCode);

cout << "Huffman Codes are:\n" << '\n';

for (auto pair: huffmanCode) {

cout << pair.first << " " << pair.second << '\n';

}

cout << "\nOriginal string was:\n" << text << '\n';

string str = "";

for (char ch: text) {

str += huffmanCode[ch];

}

cout << "\nEncoded string is:\n" << str << '\n';

cout << "\nDecoded string is:\n";

int index = -1;

while (index < (int)str.size() - 2) {

decode(root, index, str);

}

}

int main() {

string text;

cout << "Enter a string:\n";

cin >> text;

buildHuffmanTree(text);

return 0;

}