**Code:**

#include <bits/stdc++.h>

using namespace std;

struct HuffmanNode {

char data;

unsigned frequency;

HuffmanNode \*left, \*right;

HuffmanNode(char data, unsigned frequency) : data(data), frequency(frequency), left(nullptr), right(nullptr) {}

};

struct Compare {

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

return a->frequency > b->frequency;

}

};

HuffmanNode\* buildHuffmanTree(const string& text) {

unordered\_map<char, unsigned> frequencyMap;

for (char ch : text) {

frequencyMap[ch]++;

}

priority\_queue<HuffmanNode\*, vector<HuffmanNode\*>, Compare> minHeap;

for (auto& entry : frequencyMap) {

minHeap.push(new HuffmanNode(entry.first, entry.second));

}

while (minHeap.size() > 1) {

HuffmanNode\* left = minHeap.top();

minHeap.pop();

HuffmanNode\* right = minHeap.top();

minHeap.pop();

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

newNode->left = left;

newNode->right = right;

minHeap.push(newNode);

}

return minHeap.top();

}

void generateCodes(HuffmanNode\* root, string code, unordered\_map<char, string>& huffmanCodes) {

if (root) {

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

huffmanCodes[root->data] = code;

}

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

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

}

}

string encodeText(const string& text, const unordered\_map<char, string>& huffmanCodes) {

string encodedText = "";

for (char ch : text) {

encodedText += huffmanCodes.at(ch);

}

return encodedText;

}

int main() {

auto start = chrono::steady\_clock::now();

string text;

cout<<"Enter the text: ";

cin>>text;

HuffmanNode\* root = buildHuffmanTree(text);

unordered\_map<char, string> huffmanCodes;

generateCodes(root, "", huffmanCodes);

string encodedText = encodeText(text, huffmanCodes);

cout << "Huffman Codes:\n";

for (const auto& entry : huffmanCodes) {

cout << entry.first << ": " << entry.second << endl;

}

cout << "Encoded Text: " << encodedText << endl;

auto end = chrono::steady\_clock::now();

auto diff = end - start;

cout <<"Execution time: "<< chrono::duration <double, milli> (diff).count() << " ms\n";

return 0;

}

**Output:**

