NAME: Dhairya Arora

ROLL NO.- 01616401522

BTECH IT 4TH SEM 22-26

ASSIGNMENT 15

AIM: Huffman implementation

CODE:

```
#include <iostream>
#include <queue>
#include <map>
#include <string>
#include <ctime>
#include <chrono>
using namespace std;
using namespace std::chrono;

// A Huffman tree node
struct MinHeapNode {
   char data; // Input character
   unsigned freq; // Frequency of the input character
   MinHeapNode *left, *right; // Left and right child pointers

   MinHeapNode(char data, unsigned freq) {
      left = right = nullptr;
      this->data = data;
      this->freq = freq;
   }
};

// Comparison function to order the min-heap
struct compare {
```

```
    bool operator()(MinHeapNode* l, MinHeapNode* r) {
        return (l->freq > r->freq);
    }
};

// Function to print Huffman codes from the tree
void printCodes(MinHeapNode* root, string str, map<char, string>& huffmanCode) {
    if (!root)
        return;

    // Found a leaf node
    if (root->data != '$') {
        huffmanCode[root->data] = str;
    }

    printCodes(root->left, str + "0", huffmanCode);
    printCodes(root->right, str + "1", huffmanCode);
}

// Function to build the Huffman tree and print codes by traversing the tree
void buildHuffmanTree(string text) {
    map<char, unsigned> freq;
    for (char c : text) {
        freq[c]++;
    }

    priority_queue<MinHeapNode*, vector<MinHeapNode*>, compare> minHeap;

    // Create a min heap with nodes containing each character and its frequency
    for (auto pair : freq) {
        minHeap.push(new MinHeapNode(pair.first, pair.second));
```

```
    }

    // Merge nodes to create the Huffman tree
    while (minHeap.size() != 1) {
        MinHeapNode* left = minHeap.top();
        minHeap.pop();

        MinHeapNode* right = minHeap.top();
        minHeap.pop();

        MinHeapNode* top = new MinHeapNode('$', left->freq + right->freq);
        top->left = left;
        top->right = right;
        minHeap.push(top);
    }

    // Print Huffman codes
    map<char, string> huffmanCode;
    printCodes(minHeap.top(), "", huffmanCode);

    cout << "Huffman Codes are:\n" << endl;
    for (auto pair : huffmanCode) {
        cout << pair.first << " : " << pair.second << endl;
    }
}

int main() {
    string text = "hello world";
    auto start_time = high_resolution_clock::now();
    buildHuffmanTree(text);
    auto end_time = high_resolution_clock::now();
```

```
    auto duration = duration_cast<microseconds>(end_time - start_time);
    cout << "\nTime taken by Huffman Build - " << duration.count() << " Microseconds" << endl;
    return 0;
}
```

OUTPUT: