Practical N0 02: Write a program to implement Huffman Encoding using a greedy strategy.

#Code:

```
#include <iostream>
#include <vector>
#include <queue>
#include <string>
using namespace std;



struct Node {
    char symbol;
    int freq;
    Node* left;
    Node* right;

    Node(char s, int f) {
        symbol = s;
        freq = f;
        left = right = nullptr;
    }
};



struct Compare {
```

```
    bool operator()(Node* a, Node* b) {
        return a->freq > b->freq;
    }
};


void printHuffmanCodes(Node* root, string code) {
    if (!root) return;


    if (!root->left && !root->right) {
        cout << root->symbol << " -> " << code << endl;
    }

    printHuffmanCodes(root->left, code + "0");
    printHuffmanCodes(root->right, code + "1");
}

int main() {

    char chars[] = {'a', 'b', 'c', 'd', 'e', 'f'};
    int freq[] = {5, 9, 12, 13, 16, 45};
    int n = sizeof(chars) / sizeof(chars[0]);

    priority_queue<Node*, vector<Node*>, Compare> pq;
```

```
    for (int i = 0; i < n; i++) {
        pq.push(new Node(chars[i], freq[i]));
    }

    while (pq.size() > 1) {
        Node* left = pq.top(); pq.pop();
        Node* right = pq.top(); pq.pop();

        Node* parent = new Node('-', left->freq + right->freq);
        parent->left = left;
        parent->right = right;

        pq.push(parent);
    }
Node* root = pq.top();
    printHuffmanCodes(root, "");

    return 0;
}
```

#Output:

```
f -> 0
c -> 100
d -> 101
a -> 1100
b -> 1101
e -> 111
```