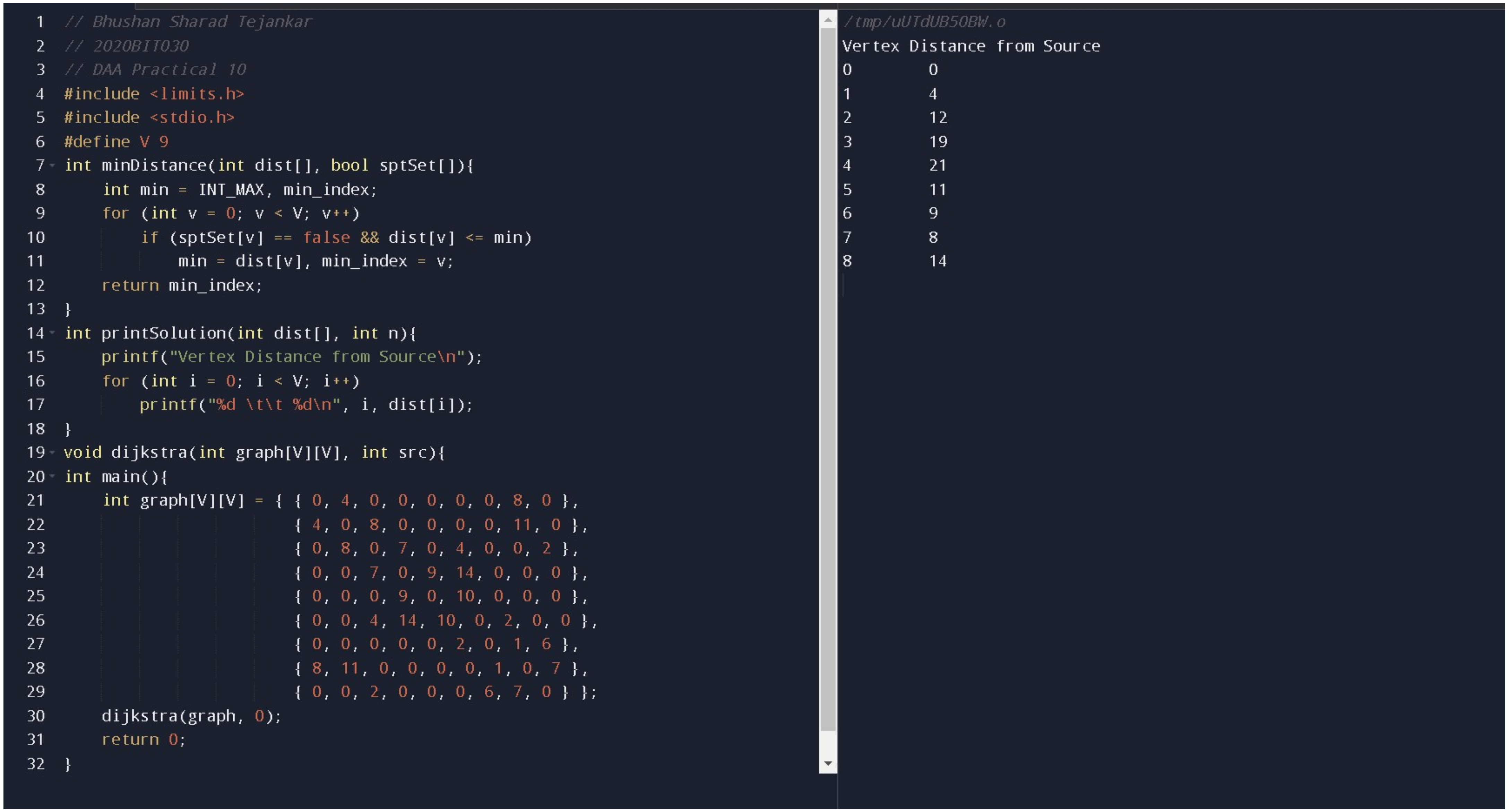
Name : Bhushan Sharad Tejankar

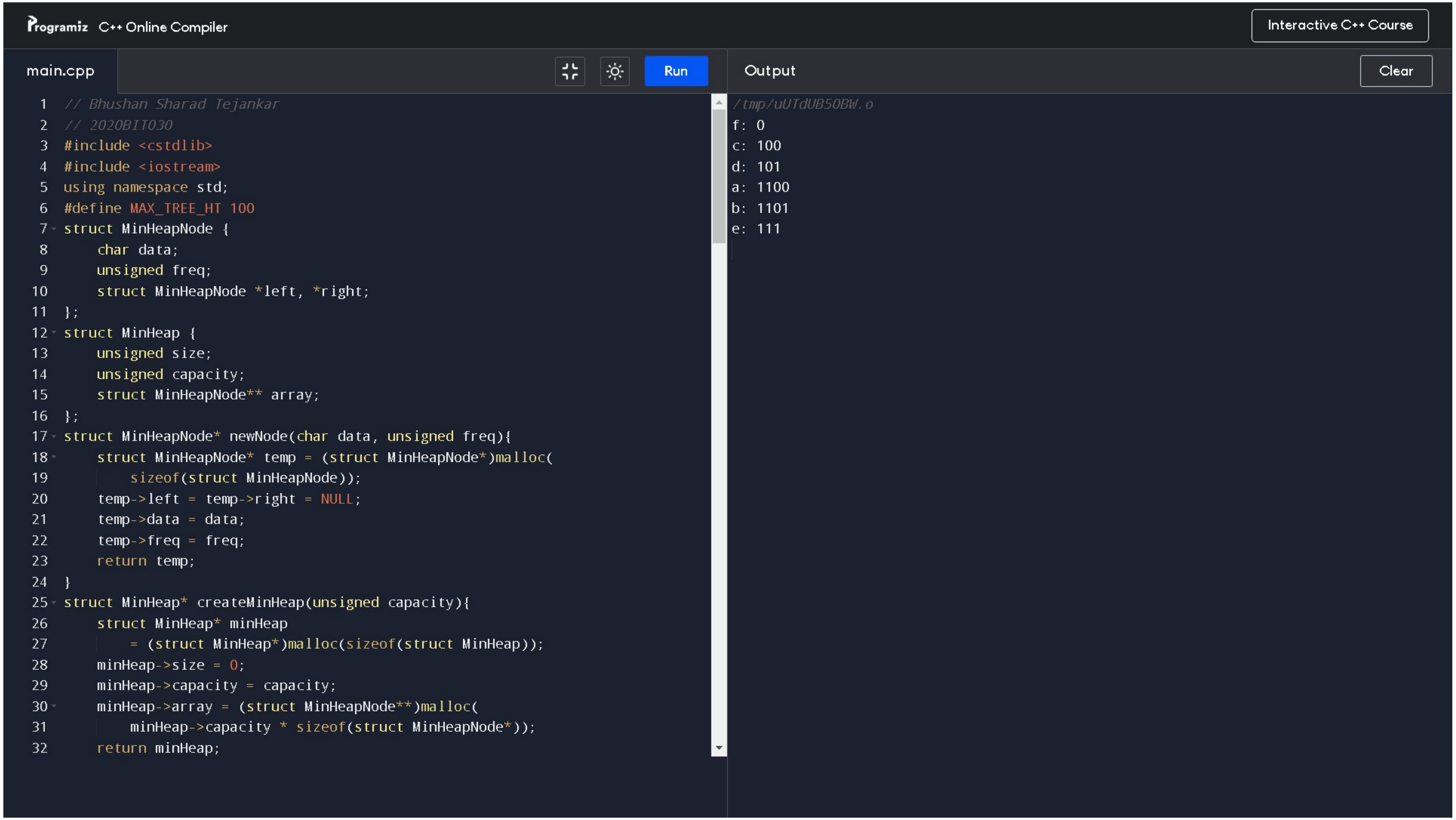
Roll no. : I30

reg_no. : 2020BIT030



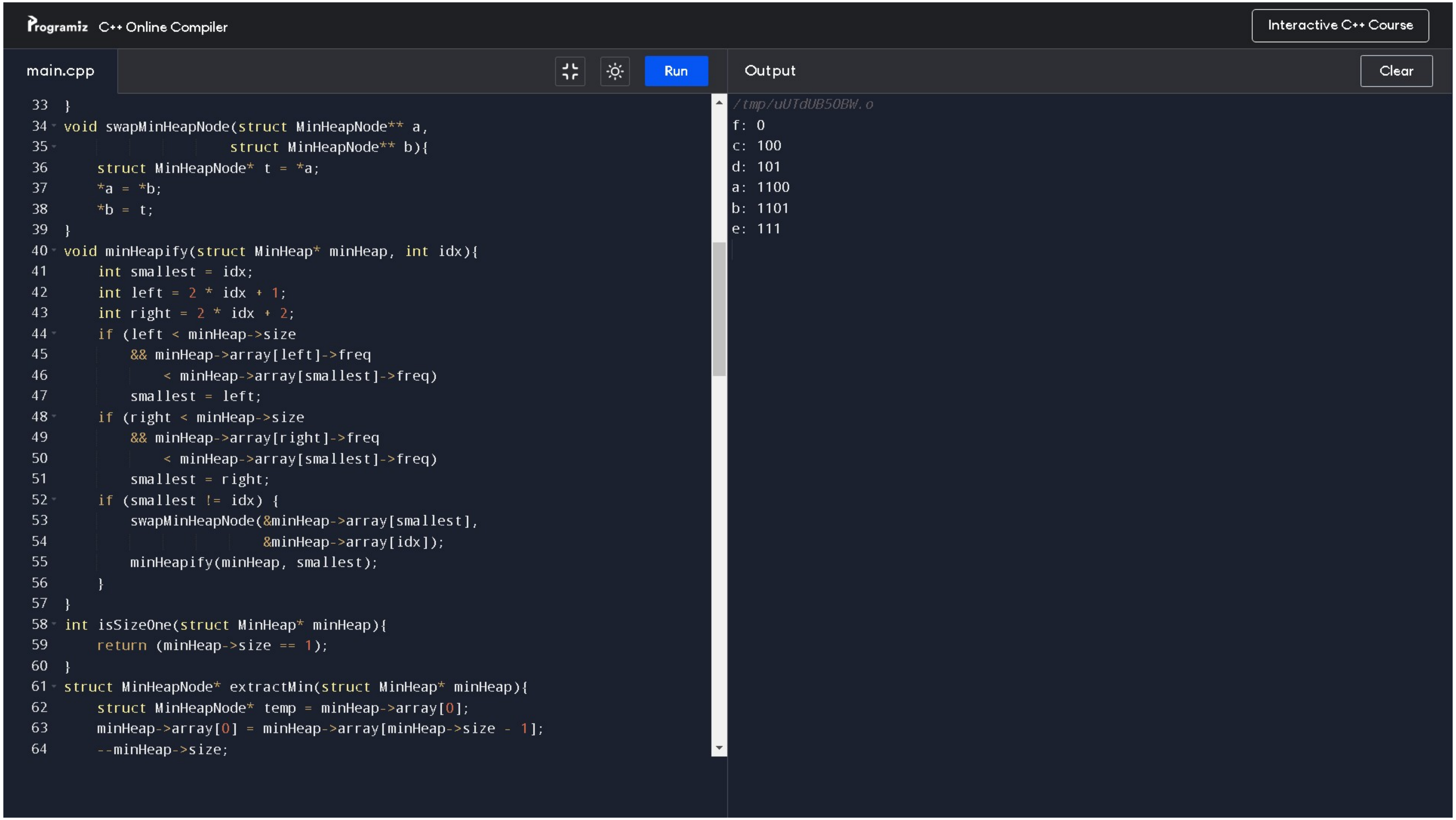
```c
// Bhushan Sharad Tejankar
// 2020BIT030
// DAA Practical 10
#include <limits.h>
#include <stdio.h>
#define V 9
int minDistance(int dist[], bool sptSet[]){
    int min = INT_MAX, min_index;
    for (int v = 0; v < V; v++)
        if (sptSet[v] == false && dist[v] <= min)
            min = dist[v], min_index = v;
    return min_index;
}
int printSolution(int dist[], int n){
    printf("Vertex Distance from Source\n");
    for (int i = 0; i < V; i++)
        printf("%d \t\t %d\n", i, dist[i]);
}
void dijkstra(int graph[V][V], int src){
int main(){
    int graph[V][V] = { { 0, 4, 0, 0, 0, 0, 0, 8, 0 },
                        { 4, 0, 8, 0, 0, 0, 0, 11, 0 },
                        { 0, 8, 0, 7, 0, 4, 0, 0, 2 },
                        { 0, 0, 7, 0, 9, 14, 0, 0, 0 },
                        { 0, 0, 0, 9, 0, 10, 0, 0, 0 },
                        { 0, 0, 4, 14, 10, 0, 2, 0, 0 },
                        { 0, 0, 0, 0, 0, 2, 0, 1, 6 },
                        { 8, 11, 0, 0, 0, 0, 1, 0, 7 },
                        { 0, 0, 2, 0, 0, 0, 6, 7, 0 } };
    dijkstra(graph, 0);
    return 0;
}
```

```
/tmp/uUTdUB50BW.o
Vertex Distance from Source
0         0
1         4
2         12
3         19
4         21
5         11
6         9
7         8
8         14
```



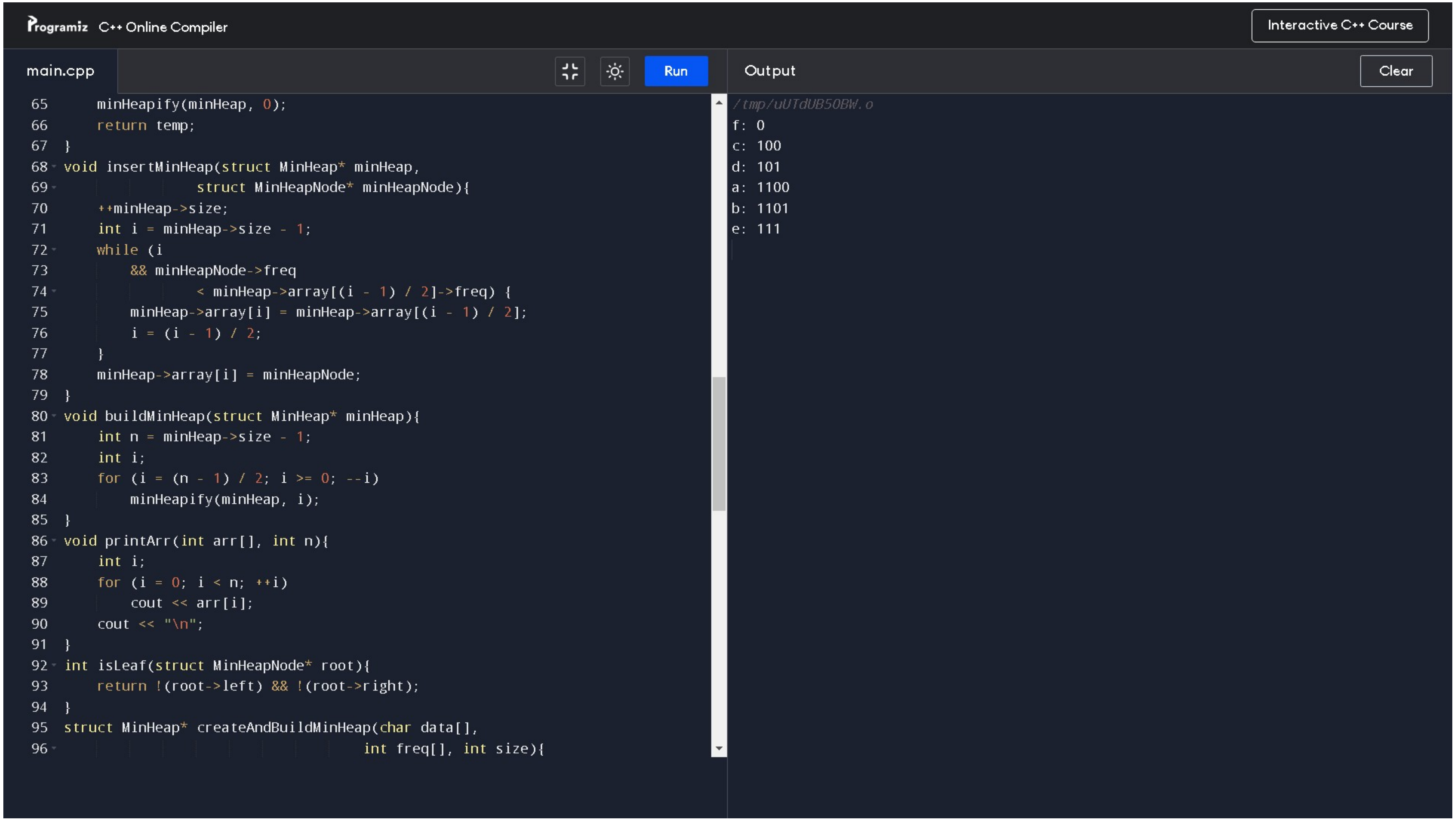
```cpp
// Bhushan Sharad Tejankar
// 2020BIT030
#include <cstdlib>
#include <iostream>
using namespace std;
#define MAX_TREE_HT 100
struct MinHeapNode {
    char data;
    unsigned freq;
    struct MinHeapNode *left, *right;
};
struct MinHeap {
    unsigned size;
    unsigned capacity;
    struct MinHeapNode** array;
};
struct MinHeapNode* newNode(char data, unsigned freq){
    struct MinHeapNode* temp = (struct MinHeapNode*)malloc(
        sizeof(struct MinHeapNode));
    temp->left = temp->right = NULL;
    temp->data = data;
    temp->freq = freq;
    return temp;
}
struct MinHeap* createMinHeap(unsigned capacity){
    struct MinHeap* minHeap
        = (struct MinHeap*)malloc(sizeof(struct MinHeap));
    minHeap->size = 0;
    minHeap->capacity = capacity;
    minHeap->array = (struct MinHeapNode**)malloc(
        minHeap->capacity * sizeof(struct MinHeapNode*));
    return minHeap;
```

```
/tmp/uUTdUB50BW.o
f: 0
c: 100
d: 101
a: 1100
b: 1101
e: 111
```



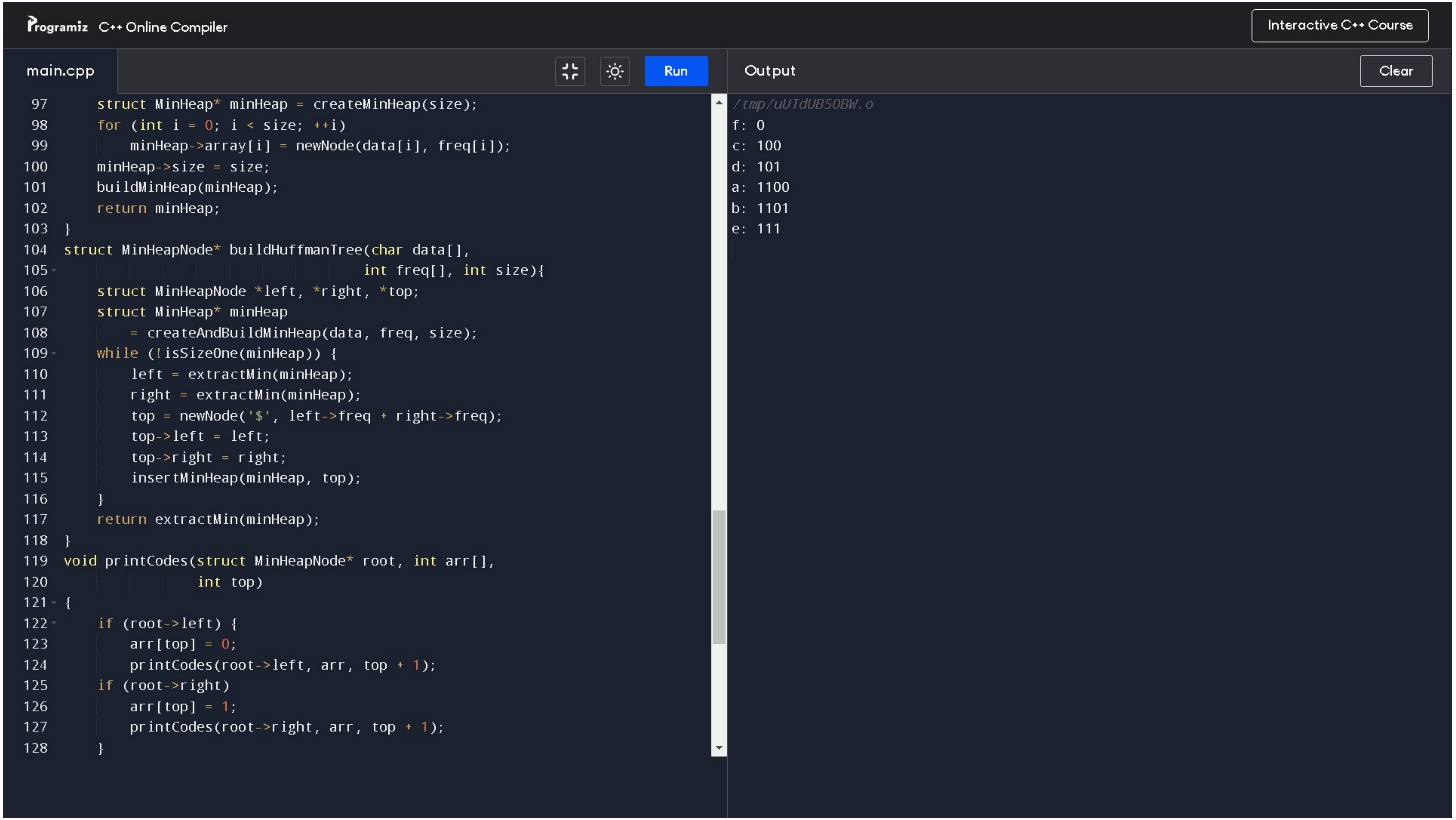
```cpp
}
void swapMinHeapNode(struct MinHeapNode** a,
                     struct MinHeapNode** b){
    struct MinHeapNode* t = *a;
    *a = *b;
    *b = t;
}
void minHeapify(struct MinHeap* minHeap, int idx){
    int smallest = idx;
    int left = 2 * idx + 1;
    int right = 2 * idx + 2;
    if (left < minHeap->size
        && minHeap->array[left]->freq
            < minHeap->array[smallest]->freq)
        smallest = left;
    if (right < minHeap->size
        && minHeap->array[right]->freq
            < minHeap->array[smallest]->freq)
        smallest = right;
    if (smallest != idx) {
        swapMinHeapNode(&minHeap->array[smallest],
                        &minHeap->array[idx]);
        minHeapify(minHeap, smallest);
    }
}
int isSizeOne(struct MinHeap* minHeap){
    return (minHeap->size == 1);
}
struct MinHeapNode* extractMin(struct MinHeap* minHeap){
    struct MinHeapNode* temp = minHeap->array[0];
    minHeap->array[0] = minHeap->array[minHeap->size - 1];
    --minHeap->size;
```

```
/tmp/uUTdUB50BW.o
f: 0
c: 100
d: 101
a: 1100
b: 1101
e: 111
```



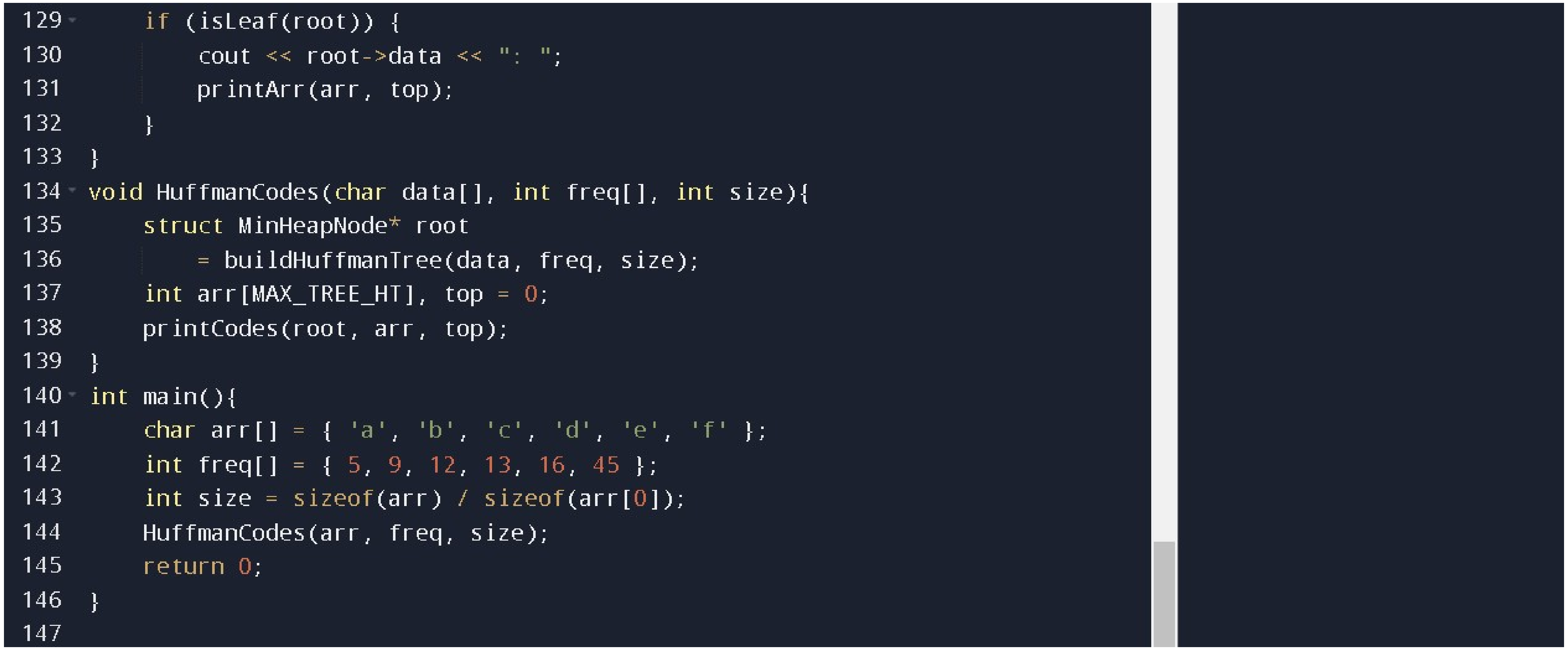
```cpp
    minHeapify(minHeap, 0);
    return temp;
}
void insertMinHeap(struct MinHeap* minHeap,
                   struct MinHeapNode* minHeapNode){
    ++minHeap->size;
    int i = minHeap->size - 1;
    while (i
        && minHeapNode->freq
                < minHeap->array[(i - 1) / 2]->freq) {
        minHeap->array[i] = minHeap->array[(i - 1) / 2];
        i = (i - 1) / 2;
    }
    minHeap->array[i] = minHeapNode;
}
void buildMinHeap(struct MinHeap* minHeap){
    int n = minHeap->size - 1;
    int i;
    for (i = (n - 1) / 2; i >= 0; --i)
        minHeapify(minHeap, i);
}
void printArr(int arr[], int n){
    int i;
    for (i = 0; i < n; ++i)
        cout << arr[i];
    cout << "\n";
}
int isLeaf(struct MinHeapNode* root){
    return !(root->left) && !(root->right);
}
struct MinHeap* createAndBuildMinHeap(char data[],
                                      int freq[], int size){
```

```
/tmp/uUTdUB5OBW.o
f: 0
c: 100
d: 101
a: 1100
b: 1101
e: 111
```



```cpp
    struct MinHeap* minHeap = createMinHeap(size);
    for (int i = 0; i < size; ++i)
        minHeap->array[i] = newNode(data[i], freq[i]);
    minHeap->size = size;
    buildMinHeap(minHeap);
    return minHeap;
}
struct MinHeapNode* buildHuffmanTree(char data[],
                                    int freq[], int size){
    struct MinHeapNode *left, *right, *top;
    struct MinHeap* minHeap
        = createAndBuildMinHeap(data, freq, size);
    while (!isSizeOne(minHeap)) {
        left = extractMin(minHeap);
        right = extractMin(minHeap);
        top = newNode('$', left->freq + right->freq);
        top->left = left;
        top->right = right;
        insertMinHeap(minHeap, top);
    }
    return extractMin(minHeap);
}
void printCodes(struct MinHeapNode* root, int arr[],
                int top)
{
    if (root->left) {
        arr[top] = 0;
        printCodes(root->left, arr, top + 1);
    if (root->right)
        arr[top] = 1;
        printCodes(root->right, arr, top + 1);
    }
```

```
/tmp/uUTdUB5OBW.o
f: 0
c: 100
d: 101
a: 1100
b: 1101
e: 111
```



```
    if (isLeaf(root)) {
        cout << root->data << ": ";
        printArr(arr, top);
    }
}
void HuffmanCodes(char data[], int freq[], int size){
    struct MinHeapNode* root
        = buildHuffmanTree(data, freq, size);
    int arr[MAX_TREE_HT], top = 0;
    printCodes(root, arr, top);
}
int main(){
    char arr[] = { 'a', 'b', 'c', 'd', 'e', 'f' };
    int freq[] = { 5, 9, 12, 13, 16, 45 };
    int size = sizeof(arr) / sizeof(arr[0]);
    HuffmanCodes(arr, freq, size);
    return 0;
}

```

THANK
YOU!