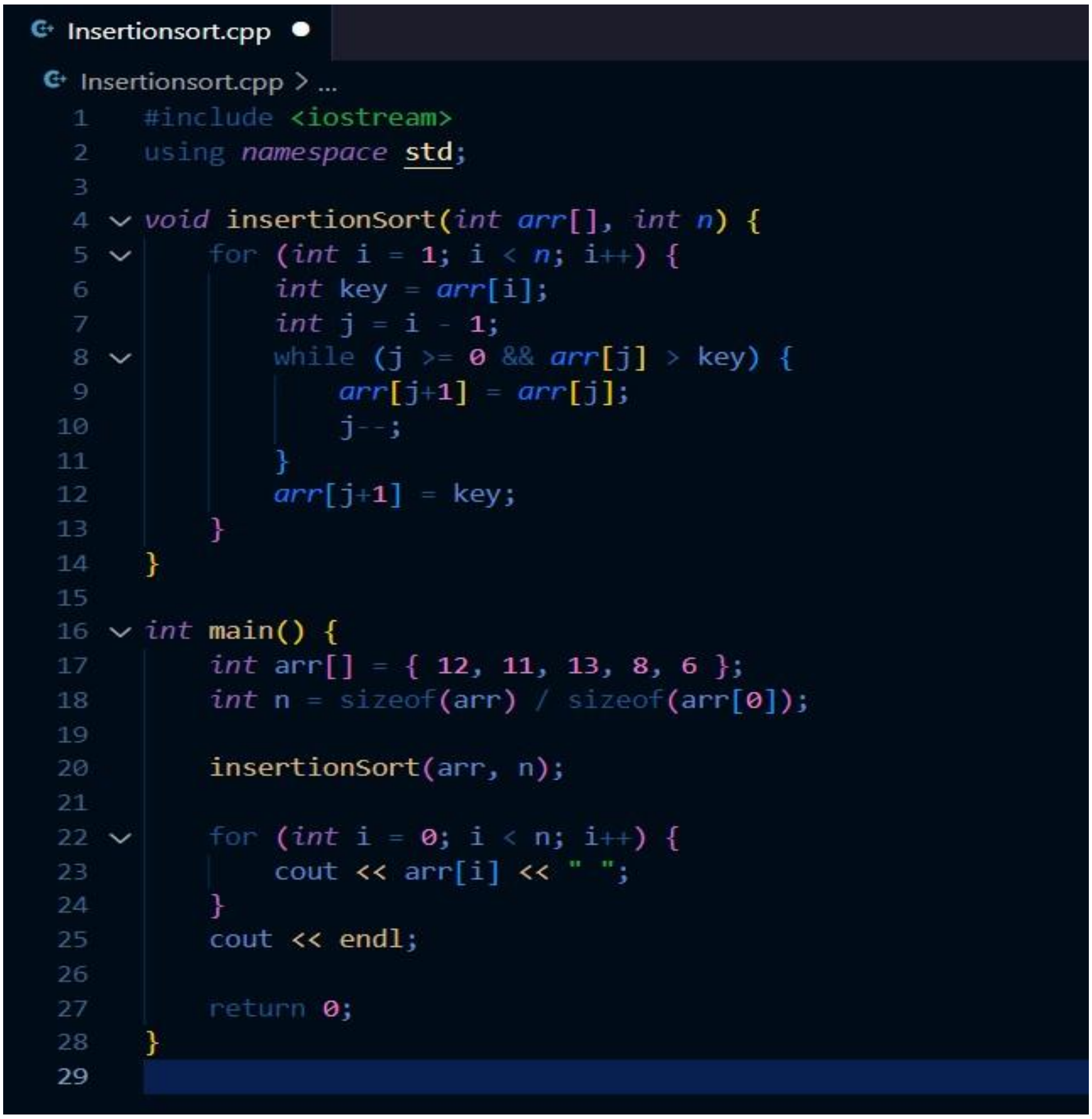
Practical-6

Name : Gaurav Ambadas Ratnaparkhi

Reg. No. : 2020BIT025

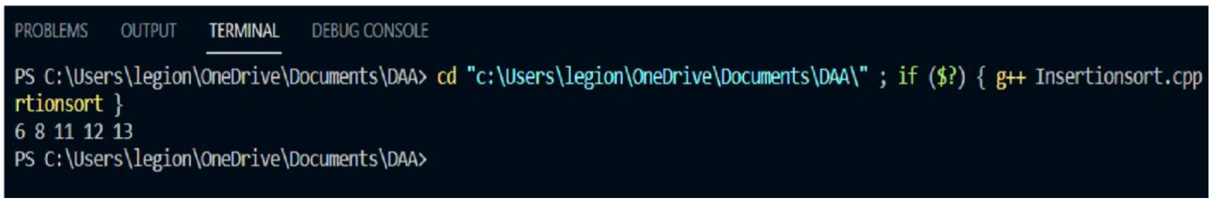
1 . Insertion Sort :

Code :



```
Insertionsort.cpp ●
Insertionsort.cpp > ...
#include <iostream>
using namespace std;

void insertionSort(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;
        while (j >= 0 && arr[j] > key) {
            arr[j+1] = arr[j];
            j--;
        }
        arr[j+1] = key;
    }
}

int main() {
    int arr[] = { 12, 11, 13, 8, 6 };
    int n = sizeof(arr) / sizeof(arr[0]);

    insertionSort(arr, n);

    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;

    return 0;
}
```

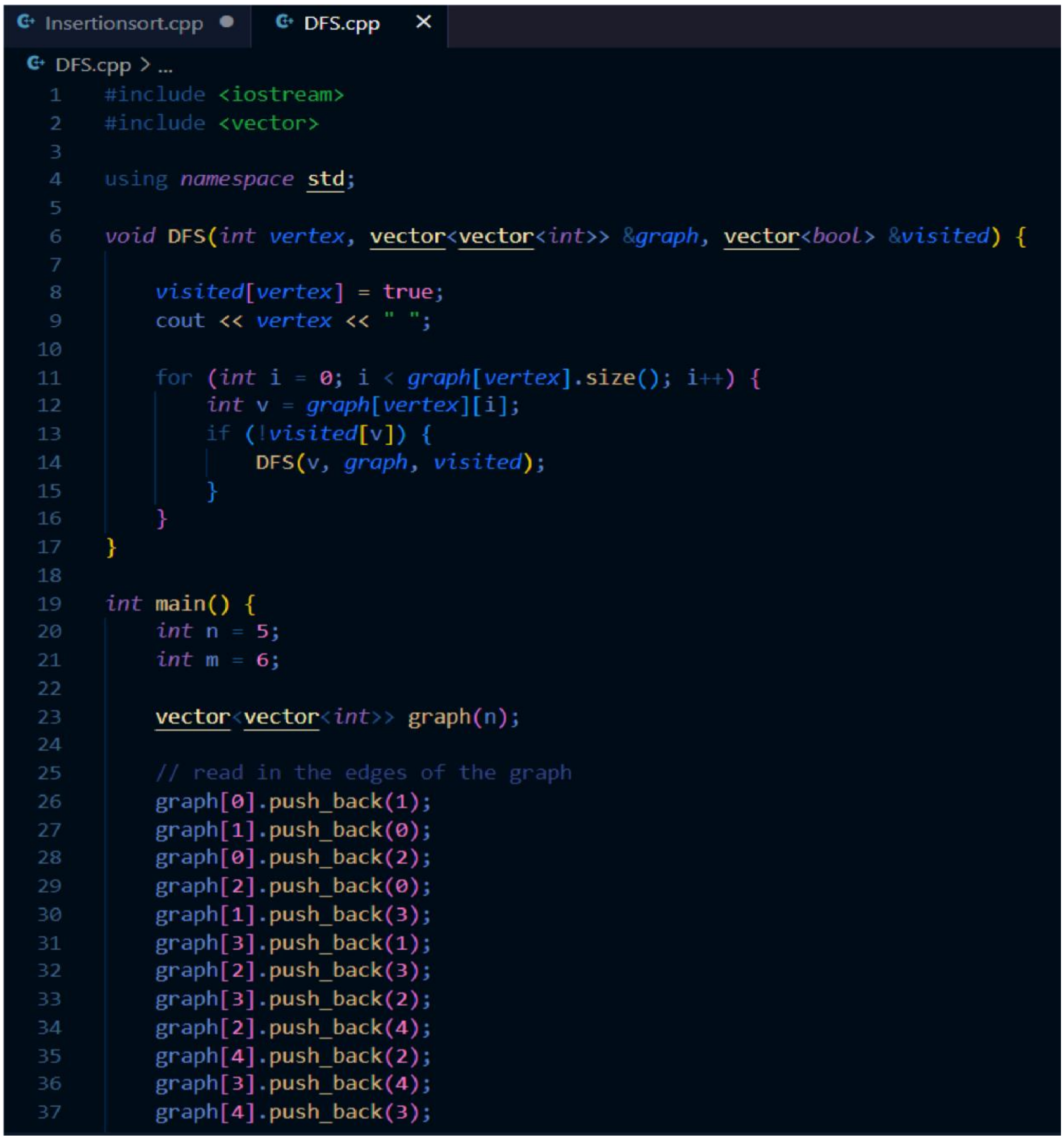
Output :



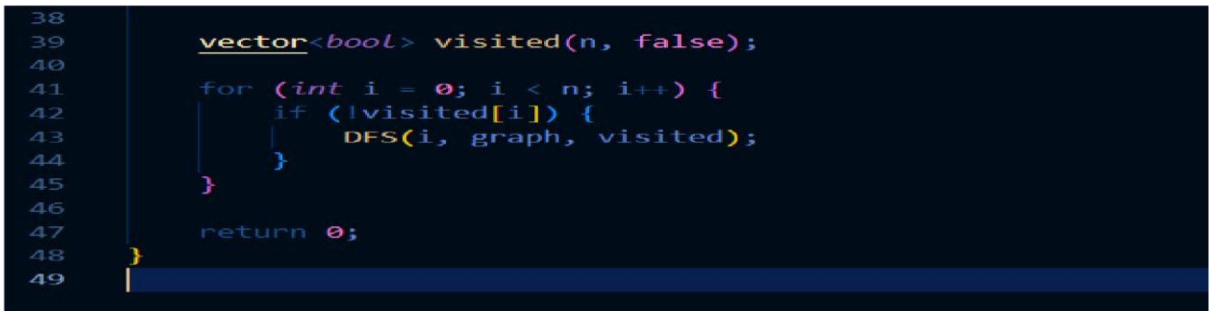
```
PS C:\Users\legion\OneDrive\Documents\DAA> cd "c:\Users\legion\OneDrive\Documents\DAA\" ; if ($?) { g++ Insertionsort.cpp
rtionsort }
6 8 11 12 13
PS C:\Users\legion\OneDrive\Documents\DAA>
```

2 . Depth First Search (DFS) :

Code :

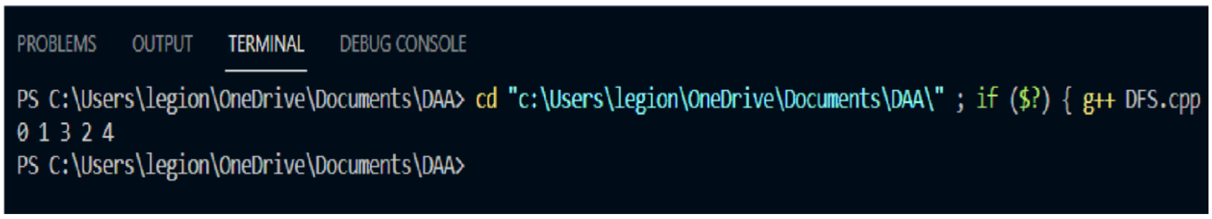


```cpp
#include <iostream>
#include <vector>

using namespace std;

void DFS(int vertex, vector<vector<int>> &graph, vector<bool> &visited) {

    visited[vertex] = true;
    cout << vertex << " ";

    for (int i = 0; i < graph[vertex].size(); i++) {
        int v = graph[vertex][i];
        if (!visited[v]) {
            DFS(v, graph, visited);
        }
    }
}

int main() {
    int n = 5;
    int m = 6;

    vector<vector<int>> graph(n);

    // read in the edges of the graph
    graph[0].push_back(1);
    graph[1].push_back(0);
    graph[0].push_back(2);
    graph[2].push_back(0);
    graph[1].push_back(3);
    graph[3].push_back(1);
    graph[2].push_back(3);
    graph[3].push_back(2);
    graph[2].push_back(4);
    graph[4].push_back(2);
    graph[3].push_back(4);
    graph[4].push_back(3);
```



```
    vector<bool> visited(n, false);

    for (int i = 0; i < n; i++) {
        if (!visited[i]) {
            DFS(i, graph, visited);
        }
    }

    return 0;
}
```

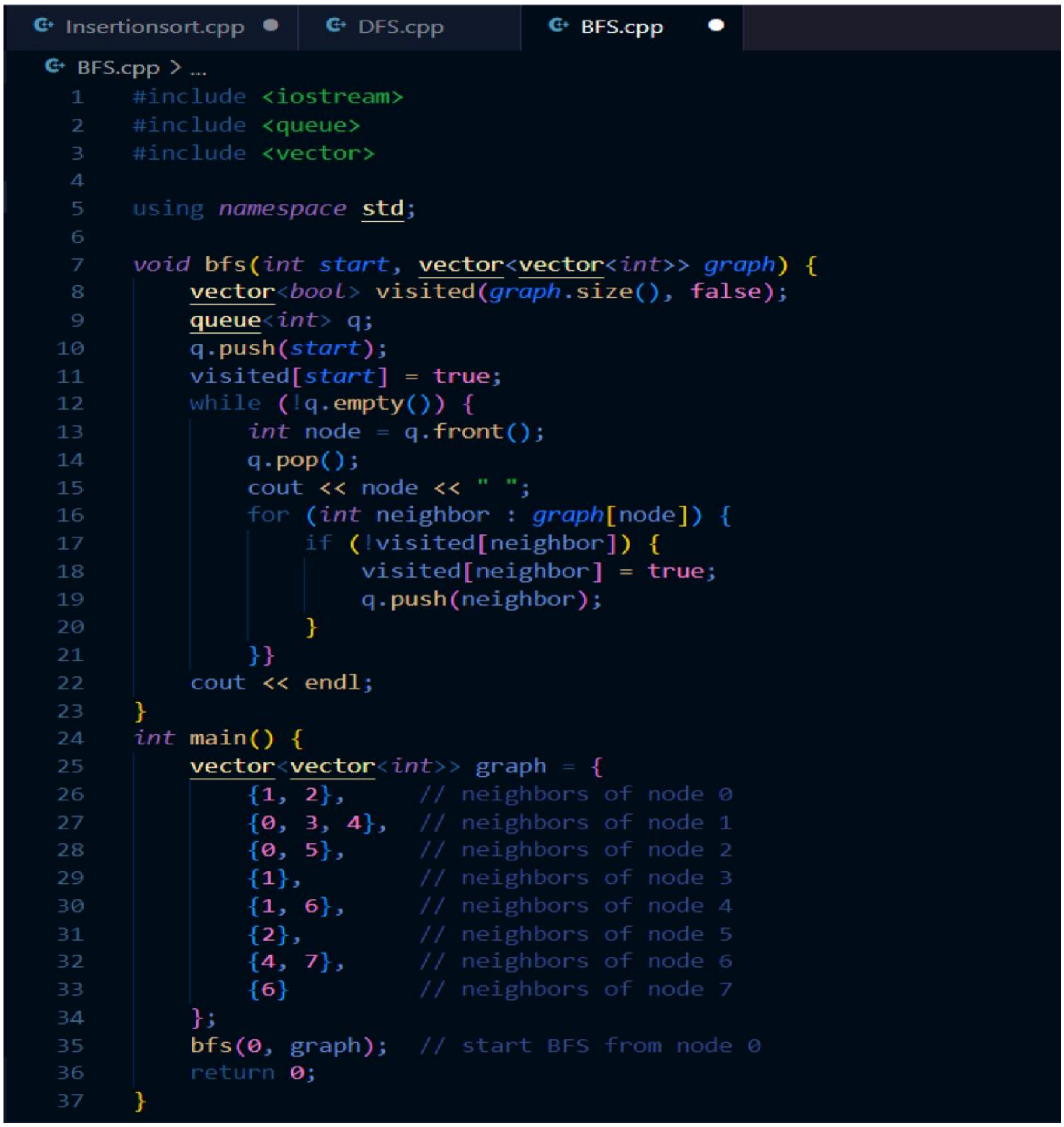
Output :



```
PS C:\Users\legion\OneDrive\Documents\DAA> cd "c:\Users\legion\OneDrive\Documents\DAA\" ; if ($?) { g++ DFS.cpp
0 1 3 2 4
PS C:\Users\legion\OneDrive\Documents\DAA>
```

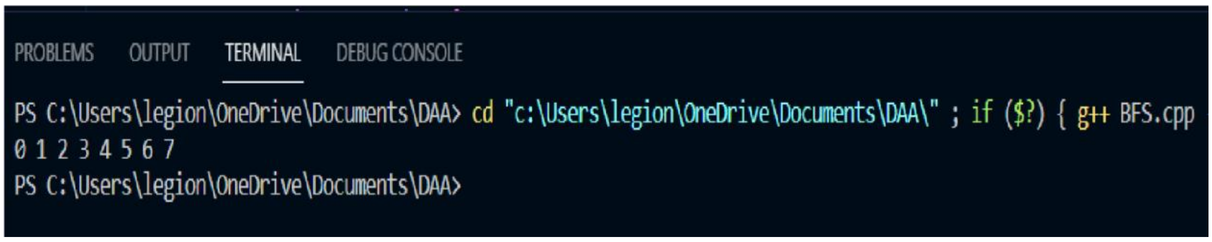
3 . Breadth First Search (BFS) :

Code :



```cpp
#include <iostream>
#include <queue>
#include <vector>

using namespace std;

void bfs(int start, vector<vector<int>> graph) {
    vector<bool> visited(graph.size(), false);
    queue<int> q;
    q.push(start);
    visited[start] = true;
    while (!q.empty()) {
        int node = q.front();
        q.pop();
        cout << node << " ";
        for (int neighbor : graph[node]) {
            if (!visited[neighbor]) {
                visited[neighbor] = true;
                q.push(neighbor);
            }
        }}
    cout << endl;
}
int main() {
    vector<vector<int>> graph = {
        {1, 2},       // neighbors of node 0
        {0, 3, 4},    // neighbors of node 1
        {0, 5},       // neighbors of node 2
        {1},          // neighbors of node 3
        {1, 6},       // neighbors of node 4
        {2},          // neighbors of node 5
        {4, 7},       // neighbors of node 6
        {6}           // neighbors of node 7
    };
    bfs(0, graph);  // start BFS from node 0
    return 0;
}
```

Output :



```
PS C:\Users\legion\OneDrive\Documents\DAA> cd "c:\Users\legion\OneDrive\Documents\DAA\" ; if ($?) { g++ BFS.cpp
0 1 2 3 4 5 6 7
PS C:\Users\legion\OneDrive\Documents\DAA>
```