Tutorial 7

Name: Shubham Santosh Dorake
PRN: B24CE1042
DIV: CE1-B

A. Breadth First Search (BFS):

Application:Indexing web pages for search engines.
Example: A web crawler uses BFS to visit web pages systematically, starting from a seed URL and exploring links level by level. Nodes represent web pages. Edges represent hyperlinks. BFS ensures that pages at the same "depth" (distance from the starting page) are visited before moving to deeper levels. Write a program to simulate the indexing of web pages for a search engine using a Breadth-First Search (BFS) algorithm.

Code :

```
#include <iostream>
using namespace std;
#define MAX 20
// Custom Queue Implementation
class Queue {
    int arr[MAX];
    int front, rear;
public:
    Queue() {
        front = -1;
        rear = -1;
    }
    bool isEmpty() {
        return (front == -1 || front > rear);
    }
```

```
    void enqueue(int x) {
        if (rear == MAX - 1) {
            cout << "Queue Overflow!" << endl;
            return;
        }
        if (front == -1)
            front = 0;
        arr[++rear] = x;
    }
    int dequeue() {
        if (isEmpty()) {
            cout << "Queue Underflow!" << endl;
            return -1;
        }
        return arr[front++];
    }
};
int main() {
    int numNodes, numEdges;
    cout << "Enter number of nodes: ";
    cin >> numNodes;
    cout << "Enter number of edges: ";
    cin >> numEdges;
    int graph[MAX][MAX] = {0};
    cout << "Enter edges (u v) where u and v are nodes (0 to " << numNodes - 1 <<
"):" << endl;
    for (int i = 0; i < numEdges; i++) {
        int u, v;
        cin >> u >> v;
        graph[u][v] = 1;
        graph[v][u] = 1; // Undirected graph
    }
    bool visited[MAX] = {false};
    Queue q;
    int startNode = 0;
```

```
    cout << "\nStarting BFS traversal from node " << startNode << ":\n";
    visited[startNode] = true;
    q.enqueue(startNode);
    while (!q.isEmpty()) {
        int node = q.dequeue();
        cout << "Visited: " << node << endl;
        for (int i = 0; i < numNodes; i++) {
            if (graph[node][i] == 1 && !visited[i]) {
                visited[i] = true;
                q.enqueue(i);
            }
        }
    }

    return 0;
}
```

B. Depth First Search (DFS):

Application: Web crawlers use DFS to explore web pages systematically, following links and indexing content for search engines. Write a simple program to index web pages using Depth First Search (DFS). The program should simulate a web graph where pages are represented as nodes and hyperlinks as edges.

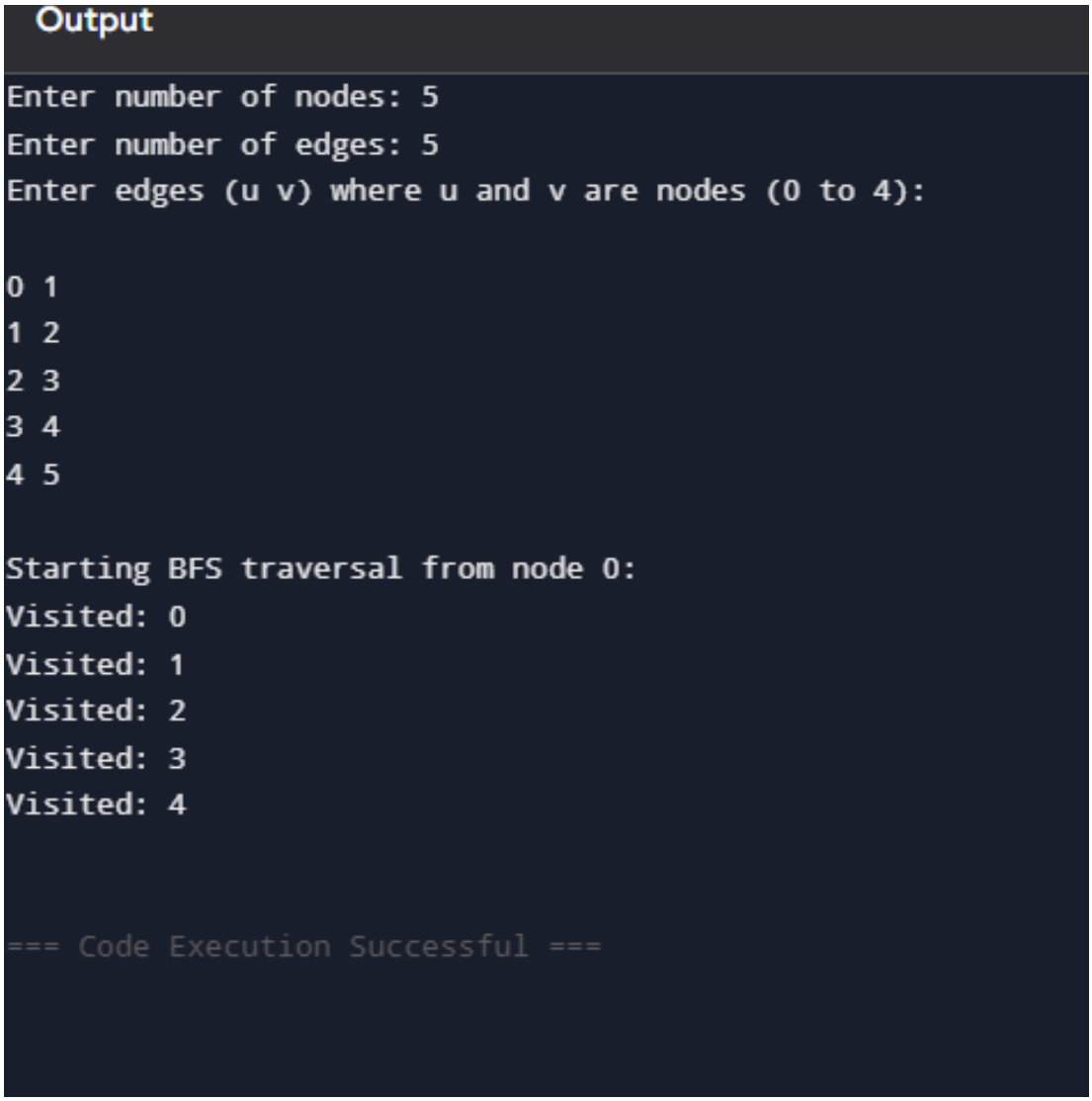
Code :

```
#include <iostream>
using namespace std;
#define MAX 20
void dfs(int node, int graph[MAX][MAX], bool visited[], int numNodes) {
    visited[node] = true;
    cout << "Visited: " << node << endl;
    for (int i = 0; i < numNodes; i++) {
```

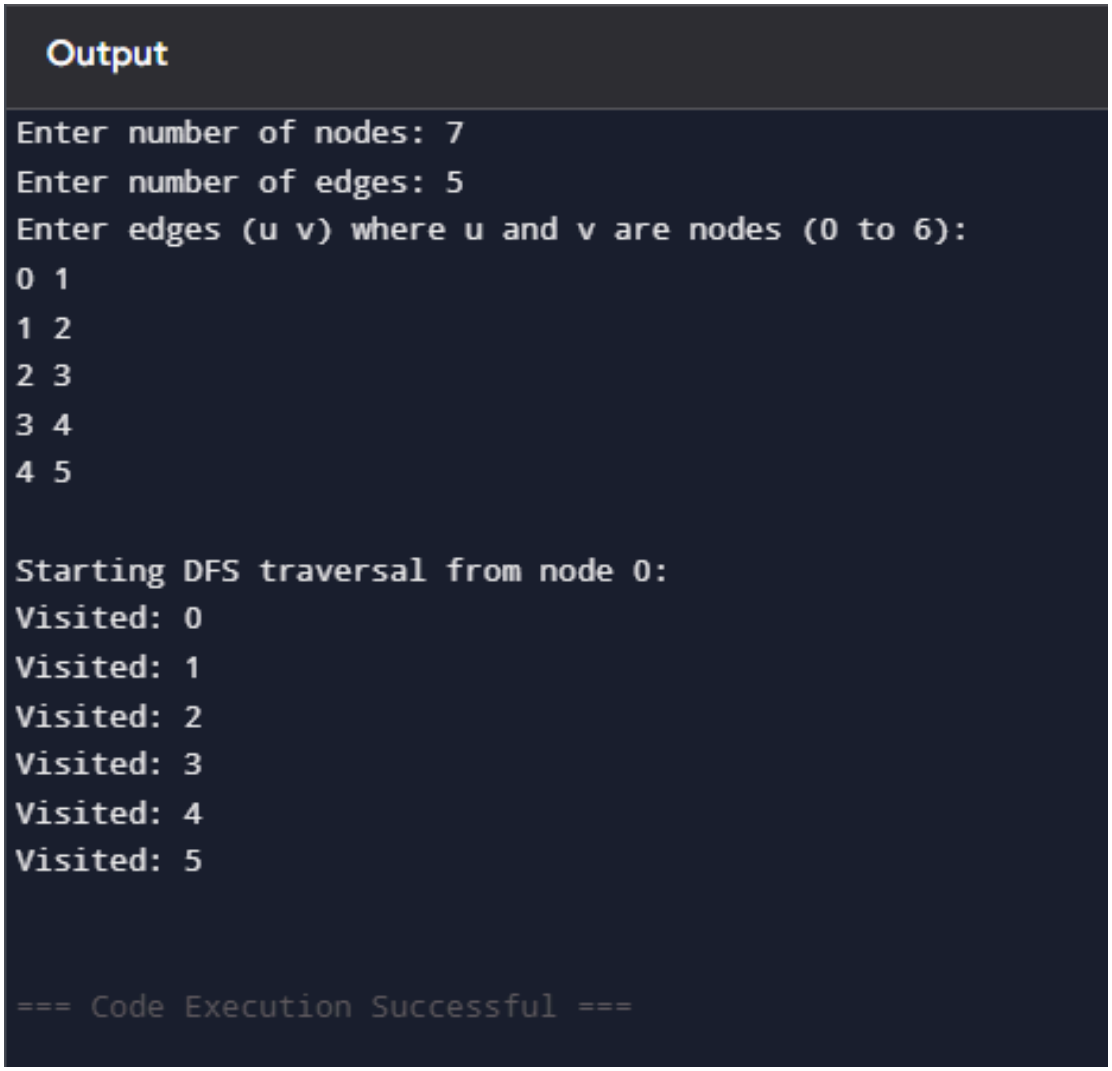
```
        if (graph[node][i] == 1 && !visited[i]) {
            dfs(i, graph, visited, numNodes);
        }
    }
}
int main() {
    int numNodes, numEdges;
    cout << "Enter number of nodes: ";
    cin >> numNodes;
    cout << "Enter number of edges: ";
    cin >> numEdges;
    int graph[MAX][MAX] = {0};
    cout << "Enter edges (u v) where u and v are nodes (0 to " << numNodes - 1 << "):"
<< endl;
    for (int i = 0; i < numEdges; ++i) {
        int u, v;
        cin >> u >> v;
        graph[u][v] = 1;
        graph[v][u] = 1; // Undirected graph
    }
    bool visited[MAX] = {false};
    cout << "\nStarting DFS traversal from node 0:\n";
    dfs(0, graph, visited, numNodes);

    return 0;
}
```

Output For BFS :



Output For DFS :



Output

```
Enter number of nodes: 7
Enter number of edges: 5
Enter edges (u v) where u and v are nodes (0 to 6):
0 1
1 2
2 3
3 4
4 5

Starting DFS traversal from node 0:
Visited: 0
Visited: 1
Visited: 2
Visited: 3
Visited: 4
Visited: 5


=== Code Execution Successful ===
```