DS PRACTICAL 7

NAME: MITHILESH YEOLE

CLASS: B3-B3-59

AIM: Write a program to reperesnt Graph in the form of adjacency Matrix.

Perform the following operations on the graph:

a. Breadth First Search

b. Depth First Search

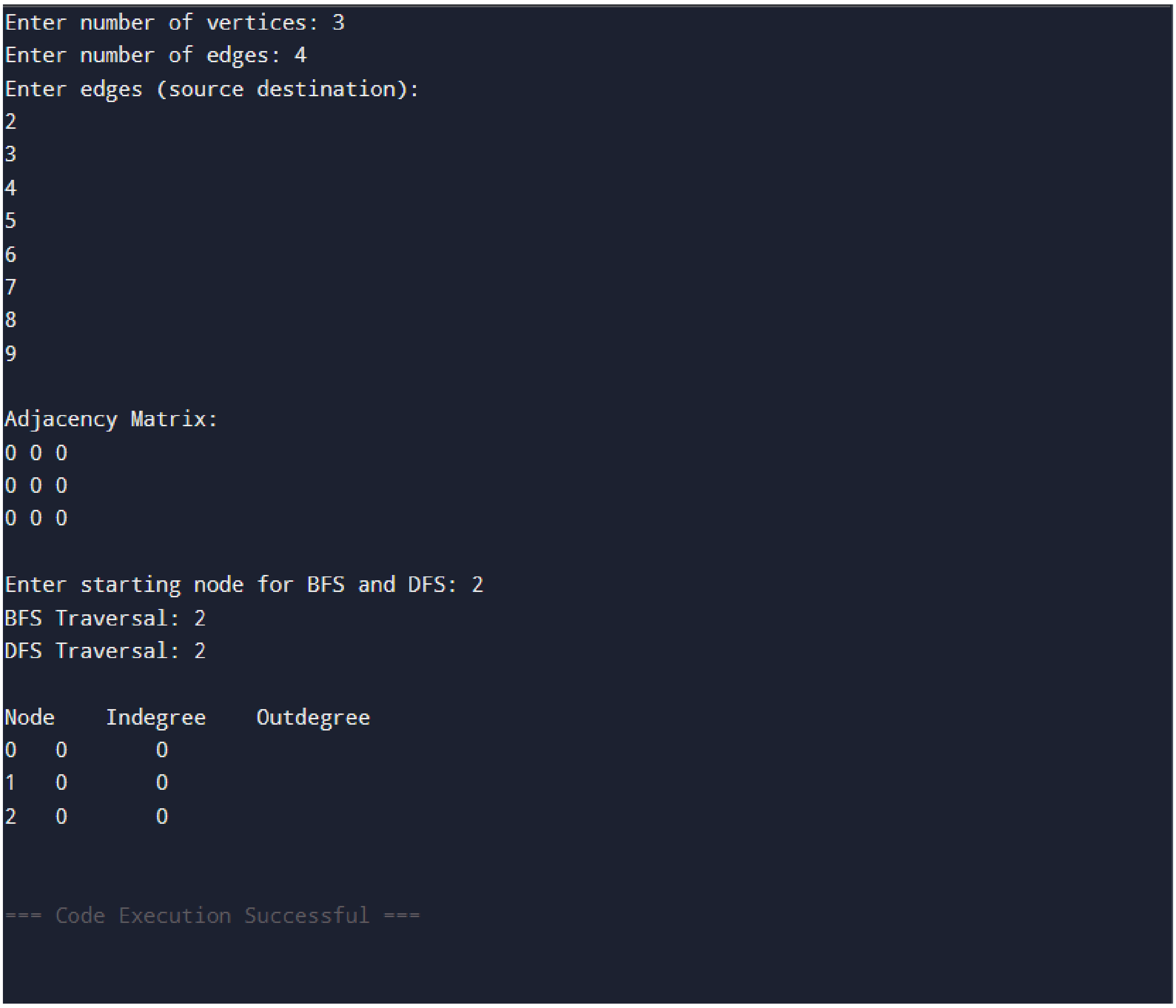
c. Indegree of a node

d. Outdegree of a node

```
#include <stdio.h>

#include <stdlib.h>


#define MAX 100


int adj[MAX][MAX];

int visited[MAX];

int queue[MAX];

int front = -1, rear = -1;


void enqueue(int value) {

    if (rear == MAX - 1)

        return;

    if (front == -1)

        front = 0;

    queue[++rear] = value;
```

```
}

int dequeue() {
    if (front == -1 || front > rear)
        return -1;
    return queue[front++];
}

int isEmpty() {
    return front == -1 || front > rear;
}

void BFS(int start, int vertices) {
    int i;
    int visitedBFS[MAX] = {0};
    front = rear = -1;
    enqueue(start);
    visitedBFS[start] = 1;

    printf("BFS Traversal: ");
    while (!isEmpty()) {
        int current = dequeue();
        printf("%d ", current);
        for (i = 0; i < vertices; i++) {
            if (adj[current][i] == 1 && visitedBFS[i] == 0) {
                enqueue(i);
                visitedBFS[i] = 1;
            }
        }
    }
    printf("\n");
}

void DFS(int node, int vertices) {
    int i;
    visited[node] = 1;
    printf("%d ", node);
    for (i = 0; i < vertices; i++) {
        if (adj[node][i] == 1 && !visited[i]) {
            DFS(i, vertices);
        }
    }
}

int indegree(int node, int vertices) {
    int count = 0;
    for (int i = 0; i < vertices; i++) {
        if (adj[i][node] == 1)
            count++;
    }
    return count;
}

int outdegree(int node, int vertices) {
    int count = 0;
    for (int i = 0; i < vertices; i++) {
        if (adj[node][i] == 1)
            count++;
    }
    return count;
}

int main() {
    int vertices, edges, i;
    int src, dest, start;

    printf("Enter number of vertices: ");
    scanf("%d", &vertices);

    printf("Enter number of edges: ");
    scanf("%d", &edges);

    printf("Enter edges (source destination):\n");
    for (i = 0; i < edges; i++) {
        scanf("%d %d", &src, &dest);
        adj[src][dest] = 1;
    }

    printf("\nAdjacency Matrix:\n");
    for (i = 0; i < vertices; i++) {
        for (int j = 0; j < vertices; j++) {
            printf("%d ", adj[i][j]);
        }
        printf("\n");
    }

    printf("\nEnter starting node for BFS and DFS: ");
    scanf("%d", &start);

    BFS(start, vertices);

    for (i = 0; i < vertices; i++) visited[i] = 0;
    printf("DFS Traversal: ");
    DFS(start, vertices);
    printf("\n");

    printf("\nNode\tIndegree\tOutdegree\n");
    for (i = 0; i < vertices; i++) {
        printf("%d\t%d\t\t%d\n", i, indegree(i, vertices), outdegree(i, vertices));
    }

    return 0;
}
```

OUTPUT:



```
Enter number of vertices: 3
Enter number of edges: 4
Enter edges (source destination):
2
3
4
5
6
7
8
9

Adjacency Matrix:
0 0 0
0 0 0
0 0 0

Enter starting node for BFS and DFS: 2
BFS Traversal: 2
DFS Traversal: 2

Node    Indegree    Outdegree
0   0       0
1   0       0
2   0       0


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