Student Information

- **Name**: [ATUL KUMAR]
- **Student ID**: [590011190]
- **Branch**: [MCA]
- **Batch**: [B1]
- **Instructor**: [Dr. Sourbh Kumar]
- **Date**: [18/11/2024]

Lab Experiment: 11 **Batch: 1 & 2**
Subject: Data Structures Lab **MCA**
Semester: 1st

Problem Statement:

Write a C program to implement graph traversal using Breadth-First Search (BFS) or Depth-First Search (DFS).

1. The program should allow the user to input the number of vertices and edges in the graph.
2. The adjacency matrix should be created based on the input.
3. Perform BFS or DFS starting from a user-defined source vertex and display the traversal order.

Assignment Tasks:

1. Input the graph's number of vertices and edges.
2. Create the graph using an adjacency matrix.
3. Implement one traversal technique (BFS or DFS).
4. Display the traversal order of the graph.

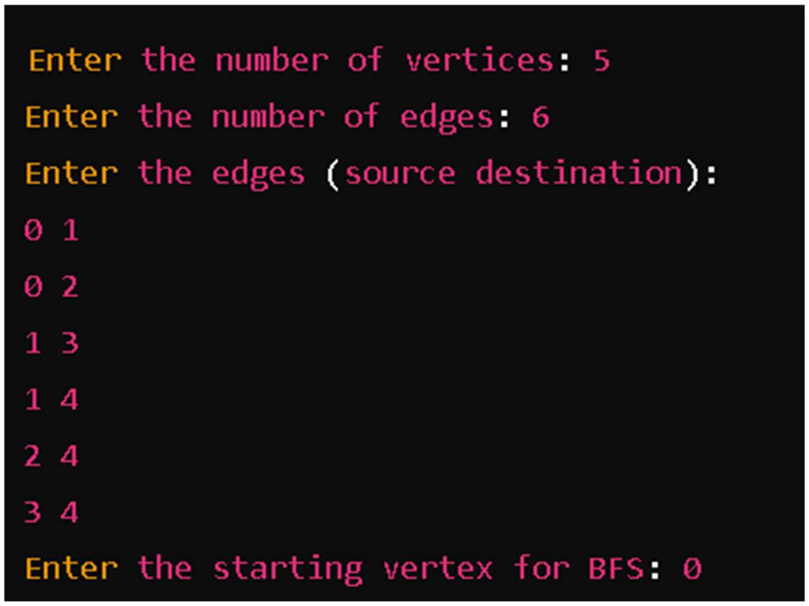
Solution:-

```
#include <stdio.h>
#include <stdlib.h>

#define MAX 100

int graph[MAX][MAX]; // Adjacency matrix
int is_visited[MAX]; // to know the visited vertices
int q[MAX]; // queue used in BFS
int front = -1, back = -1;
void add_vertex_to_queue(int vertex) {
if(back == MAX - 1) {
printf("Queue Overflow\n");
return;
}

if(front == -1) front = 0;
q[++back] = vertex;
}

int remove_vertex_from_queue() {
```

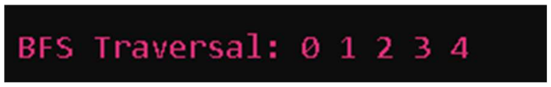
```
if(front == -1 || front > back) {
printf("Queue Underflow\n");
return -1;
}
return q[front++];
}

int check_if_empty() {
return (front == -1 || front > back);
}

void perform_BFS(int start, int vertices_count) {
for (int i = 0; i < vertices_count; i++)
is_visited[i] = 0; // All vertices are marked as not visited

printf("BFS Traversal: ");
add_vertex_to_queue(start);
is_visited[start] = 1;

while (!check_if_empty()) {
int current = remove_vertex_from_queue();
printf("%d ", current);

// Checking for all adjacent vertices
for (int i = 0; i < vertices_count; i++) {
if (graph[current][i] == 1 && is_visited[i] == 0) {
add_vertex_to_queue(i);
is_visited[i] = 1;
}
}
}
printf("\n");
}

int main() {
int num_of_vertices, num_of_edges;
printf("Enter number of vertices: ");
scanf("%d", &num_of_vertices);
printf("Enter number of edges: ");
scanf("%d", &num_of_edges);

// Initialize adjacency matrix
for (int i = 0; i < num_of_vertices; i++)
for (int j = 0; j < num_of_vertices; j++)
graph[i][j] = 0;

printf("Enter the edges (source destination):\n");
for (int i = 0; i < num_of_edges; i++) {
int fm, to;
scanf("%d %d", &fm, &to);
// Undirected graph
graph[fm][to] = 1;
graph[to][fm] = 1; // For directed graph, comment this line
}
```

```
int v;
printf("Enter the starting vertex for BFS:\n");
scanf("%d", &v);

perform_BFS(v, num_of_vertices);

return 0;
}
```

Output:-



```
Enter the number of vertices: 5
Enter the number of edges: 6
Enter the edges (source destination):
0 1
0 2
1 3
1 4
2 4
3 4
Enter the starting vertex for BFS: 0
```



```
BFS Traversal: 0 1 2 3 4
```