Saurabh kala Global id - 590014968

Lab Experiment: 11

Subject: Data Structures Lab

Objective:

To implement a graph traversal technique (Breadth-First Search or Depth-First Search) in C. This program will involve creating a graph using adjacency matrices and performing traversal on it.

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.

CODE PART –

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

#define MAX_VERTICES 50

// Function to perform DFS traversal void DFS(int graph[MAX_VERTICES][MAX_VERTICES], int
visited[], int vertex, int numVertices) {
   // Mark the current vertex as visited
visited[vertex] = 1;    printf("%d ",
vertex);

   // Recur for all the vertices adjacent to this vertex
for (int i = 0; i < numVertices; i++) {
```

```
      if (graph[vertex][i] == 1 && !visited[i]) {
         DFS(graph, visited, i, numVertices);
      }
   }
}

// Function to perform BFS traversal void BFS(int graph[MAX_VERTICES][MAX_VERTICES], int
visited[], int vertex, int numVertices) {     int queue[MAX_VERTICES], front = 0, rear = 0;

   // Mark the starting vertex as visited and enqueue it
visited[vertex] = 1;     queue[rear++] = vertex;

   while (front != rear) {
      // Dequeue a vertex from the queue
int currentVertex = queue[front++];

      // Print the dequeued vertex
printf("%d ", currentVertex);

      // Enqueue all the adjacent vertices that are not visited
for (int i = 0; i < numVertices; i++) {           if
(graph[currentVertex][i] == 1 && !visited[i]) {
            visited[i] = 1;
queue[rear++] = i;
         }
      }
   }
}

int main() {     int
graph[MAX_VERTICES][MAX_VERTICES] = {0};     int
numVertices, numEdges, choice, sourceVertex;
```

```
   // Input the number of vertices and edges
printf("Enter the number of vertices: ");
scanf("%d", &numVertices);     printf("Enter
the number of edges: ");     scanf("%d",
&numEdges);

   // Input the edges of the graph     printf("Enter the
edges (format: vertex1 vertex2):\n");     for (int i = 0; i <
numEdges; i++) {
      int u, v;
      scanf("%d %d", &u, &v);        graph[u][v] = 1;
graph[v][u] = 1; // For undirected graph, add both directions
   }

   // Input the starting vertex for BFS/DFS
printf("Enter the source vertex: ");
scanf("%d", &sourceVertex);

   // Input choice for BFS or DFS
printf("Enter 1 for BFS or 2 for DFS: ");
scanf("%d", &choice);

   // Initialize visited array     int
visited[MAX_VERTICES] = {0};

   printf("Traversal order: ");
if (choice == 1) {
      // Perform BFS
      BFS(graph, visited, sourceVertex, numVertices);
   } else if (choice == 2) {
      // Perform DFS
      DFS(graph, visited, sourceVertex, numVertices);
```

```
    } else {        printf("Invalid
choice.\n");
    }


    printf("\n");
return 0;
}
```

OUTPUT OF THE CODE

```
Enter the number of vertices: 5
Enter the number of edges: 4
Enter the edges (format: vertex1 vertex2):
0 1

0 2
1 3
2 4
3 5
Enter the source vertex: Enter 1 for BFS or 2 for DFS: 2
Traversal order: 5


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

```
Enter the number of vertices: 5
Enter the number of edges: 4
Enter the edges (format: vertex1 vertex2):
0 1
0 2
1 3
2 4
Enter the source vertex: 0
Enter 1 for BFS or 2 for DFS: 1
Traversal order: 0 1 2 3 4


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