

## Jawahar Education Society's A. C. Patil College of Engineering, Kharghar Navi Mumbai 410210

Student Name: Chetan Ingale PRN No.: 221111030

Course Name: C.S.E. (IoT CS BC)

Course code: CSL301

Year: S.E. Semester: 3

Roll No.: 17

# **Experiment Evaluation Sheet**

Experiment No.: 10

Experiment Name:
Write a program to implement graph traversal techniques.

Sr No.	Evaluation Criteria	Marks (Out of 9)	Performance Date	Correction Date and Signature of Instructor
1	<b>Experiment Performance</b>			
2	Journal Performance			
3	Punctuality			
Total				

```
Code:
```

```
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#define MAX 100
struct node {
  int vertex;
  struct node* next;
};
struct Graph {
  int numVertices;
  struct node** adjLists;
  bool* visited;
};
struct node* createNode(int v) {
  struct node* newNode = (struct node*)malloc(sizeof(struct node));
  newNode->vertex = v;
  newNode->next = NULL;
  return newNode;
struct Graph* createGraph(int vertices) {
  struct Graph* graph = (struct Graph*)malloc(sizeof(struct Graph));
  graph->numVertices = vertices;
  graph->adjLists = (struct node**)malloc(vertices * sizeof(struct node*));
  graph->visited = (bool*)malloc(vertices * sizeof(bool));
  for (int i = 0; i < vertices; i++) {
    graph->adjLists[i] = NULL;
    graph->visited[i] = false;
  return graph;
void addEdge(struct Graph* graph, int src, int dest) {
  struct node* newNode = createNode(dest);
  newNode->next = graph->adjLists[src];
  graph->adjLists[src] = newNode;
  newNode = createNode(src);
  newNode->next = graph->adjLists[dest];
  graph->adjLists[dest] = newNode;
}
void DFS(struct Graph* graph, int vertex) {
  graph->visited[vertex] = true;
  printf("Visited %d\n", vertex);
  struct node* adjList = graph->adjLists[vertex];
  while (adjList != NULL) {
```

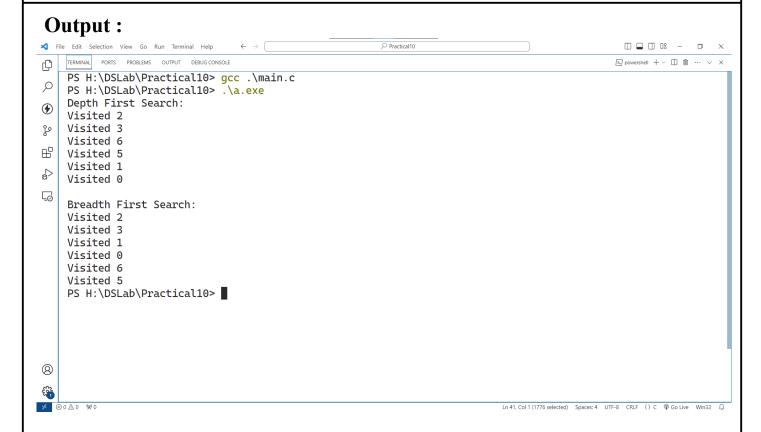
Name: Chetan Ingale Roll No.: 17 Page No.:2

```
Code:
      if (!graph->visited[adjList->vertex]) {
        DFS(graph, adjList->vertex);
      adjList = adjList->next;
 void BFS(struct Graph* graph, int startVertex) {
   bool* visited = (bool*)malloc(graph->numVertices * sizeof(bool));
   for (int i = 0; i < graph->numVertices; i++) {
      visited[i] = false;
   int queue[MAX];
   int front = 0, rear = 0;
   queue[rear] = startVertex;
   rear++;
   visited[startVertex] = true;
   while (front < rear) {
      int currentVertex = queue[front];
      printf("Visited %d\n", currentVertex);
      front++;
      struct node* temp = graph->adjLists[currentVertex];
      while (temp) {
        int adjVertex = temp->vertex;
        if (!visited[adjVertex]) {
           queue[rear] = adjVertex;
           rear++;
           visited[adjVertex] = true;
        temp = temp->next;
int main() {
   struct Graph* graph = createGraph(7);
   addEdge(graph, 0, 1);
   addEdge(graph, 0, 2);
   addEdge(graph, 1, 2);
   addEdge(graph, 2, 3);
   addEdge(graph, 5, 3);
   addEdge(graph, 6, 3);
   addEdge(graph, 5, 6);
   printf("Depth First Search:\n");
   DFS(graph, 2);
   printf("\nBreadth First Search:\n");
   BFS(graph, 2);
   return 0;
```

Name: Chetan Ingale Roll No.: 17 Page No.:3

### A. C. Patil College of Engineering

#### **Data Structure Lab**



#### **Conclusion:**

Through this experiment we have learnt about how to implement a graph using the C language. Various operations like insertion and traversal are applied on the graph.

This experiment helps us in using graph as a data structure for further reference.

Name: Chetan Ingale Roll No.: 17 Page No.:4