COMPUTER LAB

NAME: SOUMYADIP MAITY

ROLL NO.: 22053029

SEC: CSE-49

YEAR: 2023-24

ASSIGNMENT – 12

1.WAP to create an un-directed Graph using Adjacency Matrix.

```
#include <stdio.h>
#define MAX_VERTICES 100
int adjacencyMatrix[MAX_VERTICES][MAX_VERTICES];
int numVertices;
void createGraph() {
int i, j;
printf("Enter the numberof vertices: ");
scanf("%d", &numVertices);
for (i = 0; i < numVertices; i++) {
for (j = 0; j < numVertices; j++) {
adjacencyMatrix[i][j] = 0;
}
}
for (i = 0; i < numVertices; i++) {
for (j = i + 1; j < numVertices; j++) {
int edge;
printf("Is there an edge betweenvertex %d andvertex %d? (1 for Yes, 0 for No): ", i, j);
scanf("%d", &edge);
adjacencyMatrix[i][j] = edge;
adjacencyMatrix[j][i] = edge;
}
}
}
void displayGraph() {
int i, j;
printf("Adjacency Matrix:\n");
for (i = 0; i < numVertices; i++) {
for (j = 0; j < numVertices; j++) {
printf("%d", adjacencyMatrix[i][j]);
}
printf("\n");
}
}
int main() {
createGraph();
displayGraph();
return 0;
}
```

2. Modify the Above Program to include a menu-driven Program and add options for the depath-first traversal and breadth-first traversal and Write code for those traversals.

```
#include <stdio.h>
#include <stdlib.h>
#define MAX_VERTICES 100
struct Graph{
```

```
int numVertices;
int adjacencyMatrix[MAX_VERTICES][MAX_VERTICES];
};
void initializeGraph(struct Graph*graph, int numVertices) {
graph->numVertices = numVertices;
for (int i = 0; i < numVertices; i++) {
for (int j = 0; j < numVertices; j++) {
graph->adjacencyMatrix[i][j] = 0;
}
}
}
void addEdge(struct Graph*graph, int source, int destination) {
graph->adjacencyMatrix[source][destination] = 1;
graph->adjacencyMatrix[destination][source] = 1;
}
void depthFirstTraversal(struct Graph*graph, int vertex, int visited[]) {
visited[vertex] = 1;
printf("%d ", vertex);
for (int i = 0; i < graph->numVertices; i++) {
if (graph->adjacencyMatrix[vertex][i] == 1 && visited[i] == 0) {
depthFirstTraversal(graph, i, visited);
}
}
}
void breadthFirstTraversal(struct Graph*graph, int startVertex) {
int visited[MAX_VERTICES] = {0};
int queue[MAX_VERTICES];
int front = 0, rear = 0;
visited[startVertex] = 1;
printf("%d ", startVertex);
queue[rear++] = startVertex;
while (front < rear) {
int currentVertex = queue[front++];for (int i = 0; i < graph->numVertices; i++) {
if (graph->adjacencyMatrix[currentVertex][i] == 1 && visited[i] == 0) {
visited[i] = 1;
printf("%d ", i);
queue[rear++] = i;
}
}
}
}
int main() {
struct Graph graph;
int numVertices,choice, source,destination;
printf("Enter the numberof vertices in the graph: ");
scanf("%d", &numVertices);
initializeGraph(&graph, numVertices);
while (1) {
printf("\nMenu:\n");
printf("1. Addan edge\n");
printf("2. Perform depth-first traversal\n");
```

```
printf("3. Perform breadth-first traversal\n");
printf("4. Exit\n");
printf("Enter your choice: ");
scanf("%d", &choice);
switch(choice) {
case1:
printf("Enter the sourceand destinationvertices: ");
scanf("%d %d", &source, &destination);
addEdge(&graph, source,destination);
break;
case2:
printf("Depth-first traversal: ");
int visited[MAX_VERTICES] = {0};
for (int i = 0; i < graph.numVertices; i++) {
if (visited[i] == 0) {
depthFirstTraversal(&graph, i, visited);
}
}
printf("\n");
break;
case 3:
printf("Breadth-first traversal: ");
breadthFirstTraversal(&graph, 0);printf("\n");
break;
case4:
exit(0);
default:
printf("Invalid choice. Pleasetry again.\n");
}
}
return 0;
}
```

3. WAP to create a directed graph using the adjacency Matrix Method.

```
#include <stdio.h>
#define MAX_VERTICES 100
int graph [MAX_VERTICES] [MAX_VERTICES];
void createDirectedGraph(int vertices)
{
int i, j;
for (i = 0; i < vertices; i++)
{
for (j = 0; j < vertices; j++)
{
graph[i][j] = 0;
}
}
```

```
int source,destination;
while (1)
{
printf("Enter source vertex (-1 to exit): ");
scanf("%d", &source);
if (source == -1)
break;
printf("Enterdestination vertex: ");
scanf("%d", &destination);
if (source>= vertices || destination >= vertices || source< 0 || destination < 0)
{
printf("Invalid vertices! Pleasetry again.\n");
continue;
}
graph[source][destination] = 1;
}
}
void displayGraph(int vertices)
{
int i, j;
printf("Directed Graph:\n");
for (i = 0; i < vertices; i++)
{
for (j = 0; j < vertices; j++){
printf("%d", graph[i][j]);
}
printf("\n");
}
}
int main()
{
int vertices;
printf("Enter the numberof vertices in the graph: ");
scanf("%d", &vertices);
createDirectedGraph(vertices);
displayGraph(vertices);
return 0;
}
```