**Data Structures Lab [BCSL305]**

**Program 11**

**Design, Develop and Implement a Program in C for the following operations on Graph(G) of Cities a. Create a Graph of N cities using Adjacency Matrix.**

**b. Print all the nodes reachable from a given starting node in a digraph using BFS/DFS method.**

#include <stdio.h>

#define MAX\_CITIES 20

int adjMatrix[MAX\_CITIES][MAX\_CITIES]; // Adjacency matrix to store the graph

int visited[MAX\_CITIES]; // Array to keep track of visited nodes

int n; // Number of cities (vertices)

void createGraph() {

int i, j;

printf("Enter the number of cities: ");

scanf("%d", &n);

// Initialize the adjacency matrix with zeros

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++) {

adjMatrix[i][j] = 0;

}

}

printf("Enter the adjacency matrix (1 for edge, 0 for no edge):\n");

for (i = 0; i < n; i++) {

for (j = 0; j < n; j++) {

scanf("%d", &adjMatrix[i][j]);

}

}

}

void bfs(int start) {

int queue[MAX\_CITIES], front = -1, rear = -1, i;

// Initialize visited array

for (i = 0; i < n; i++) {

visited[i] = 0;

}

// Start BFS from the given node

visited[start] = 1;

queue[++rear] = start;

printf("BFS starting from city %d: ", start + 1); // Cities are numbered 1 to n

while (front != rear) {

int current = queue[++front];

printf("%d ", current + 1); // Print city number (1-based index)

// Visit all neighbors of the current city

for (i = 0; i < n; i++) {

if (adjMatrix[current][i] == 1 && visited[i] == 0) {

visited[i] = 1;

queue[++rear] = i;

}

}

}

printf("\n");

}

void dfs(int start) {

int stack[MAX\_CITIES], top = -1, i;

// Initialize visited array

for (i = 0; i < n; i++) {

visited[i] = 0;

}

// Start DFS from the given node

visited[start] = 1;

stack[++top] = start;

printf("DFS starting from city %d: ", start + 1); // Cities are numbered 1 to n

while (top != -1) {

int current = stack[top--];

printf("%d ", current + 1); // Print city number (1-based index)

// Visit all neighbors of the current city

for (i = 0; i < n; i++) {

if (adjMatrix[current][i] == 1 && visited[i] == 0) {

visited[i] = 1;

stack[++top] = i;

}

}

}

printf("\n");

}

int main() {

int choice, startCity;

do {

printf("\nMenu:\n");

printf("1. Create Graph\n");

printf("2. BFS Traversal\n");

printf("3. DFS Traversal\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

createGraph();

break;

case 2:

printf("Enter the starting city for BFS (1 to %d): ", n);

scanf("%d", &startCity);

startCity--; // Convert to 0-based index

if (startCity >= 0 && startCity < n) {

bfs(startCity);

} else {

printf("Invalid city number.\n");

}

break;

case 3:

printf("Enter the starting city for DFS (1 to %d): ", n);

scanf("%d", &startCity);

startCity--; // Convert to 0-based index

if (startCity >= 0 && startCity < n) {

dfs(startCity);

} else {

printf("Invalid city number.\n");

}

break;

case 4:

printf("Exiting program.\n");

break;

default:

printf("Invalid choice. Please try again.\n");

break;

}

} while (choice != 4);

return 0;

}