National University of Computer and Emerging Sciences

Lab Manual

for

Data Structure

Course Instructor	Ms. Abeeda Akram
Lab Instructor(s)	Mr. Sohaib Ahmad Ms. Ammara
Section	BCS-3G
Semester	FALL 2022

Department of Computer Science
FAST-NU, Lahore, Pakistan

Lab Manual 13

Objectives:

After performing this lab, students shall be able to revise:

✔ Graphs
✔ BFS
✔ DFS

Question 1

You are required to implement the adjacency list class. Assume the graph is directed.

```
#include<iostream>

using namespace std;

class Graph

{

    int V; // No. of vertices

    int **adj; // adjacency matrix

    bool isDirected; // 0 for undirected, 1 for directed

    Graph(int n); // Constructor

    void TakeInput(int v, int w); // add an edge to the graph
    void DFS(int start); // print the paths from start to every other vertex as generated by the DFS method
    void BFS(int start); // print the paths from start to every other vertex using BFS method.
    bool Is_Connected(int n); // returns true if graph is connected using DFS and BFS both
    bool isReachable(int u, int v); // Check and returns true if there is a path between any two given vertices.
    bool is_SC(); A directed graph is said to be strongly connected if every vertex is reachable from every other
vertex. Returns true if it is strongly connected otherwise false?
    bool isCylic();  // returns true if there is a cycle in the graph for directed and undirected graph
};
```

Write a proper main to test all of these functions.

Question 2 (BONUS)

There are *N* cities connected by *M* bidirectional roads. Each of these cities has an Airport which can be used to travel from one to another city regardless of whether these are connected by a road or not. Your task is to help your friend visit all these cities using at most *K* flights.

Provide a sequence which makes your friend happy. If It's not possible to do so, print −1. Your friend is okay if he revisits a city, but he can't afford more than K flights

Input:
The first line contains three integers *N, M, K;* the number of cities, the number of roads and the maximum number of flights he can take.
Next *M* lines contains two integers *x* and *y*, denoting a road between city *x* and *y*.

Output:
Output *N* numbers separated by spaces, which is the order of all the cities to be visited. If it's not possible, output −1.

Sample Input	Sample Output

5 3 2
1 2
3 4
4 5
	1 2 3 4 5

Explanation:

This sequence can be achieved if your friend starts from city number 1 and takes the only flight from 2 – 3.