**National University of Computer and Emerging Sciences**



**Laboratory Manual**

*for*

**Data Structures Lab 13**

|  |  |
| --- | --- |
| Course Instructor | Mr. Uzair Naqvi |
| Lab Instructor(s) | Fariha Maqbool  Marwa Khan |
| Section | BCS-4H |
| Semester | Fall 2022 |

**Department of Computer Science**

FAST-NU, Lahore, Pakistan

**Objectives:**

In this lab, students will practice:

1. Graphs
2. DFS
3. BFS

**Question 1**

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

#include<iostream> #include<List> using namespace std; class Graph

{

int v; // No. of vertices list<int> \*adj; // adjacency lists public:

Graph(int n); // Constructor

void TakeInput(int n, int w); // to take input from the user in this sequence: number of

nodes, what are the neighbors of vertex 0, what are the neighbors of vertex 1, ... so on.

void ExploreFunction(int start); // print the paths from start to every other vertex as

generated by the dfs method. One path per line.

};

int main()

{

Graph g(4); // Total 5 vertices in graph

g.TakeInput(0, 1);

g.TakeInput(0, 2);

g.TakeInput(1, 2);

g.TakeInput(2, 0);

g.TakeInput(2, 3);

g.TakeInput(3, 3); cout << "Following is Depth First Traversal\n";

g.ExploreFunction(2);

//Sample output of explore function for starting from vertex 2.

//2 0

//2 0 1

//2 0 1 3

//\*\*\*\*\*\*\*\*\*

//Sample output of explore function for starting from vertex 1.

//1 2

// 1 2 0 //1 2 0 3

return 0;

}

**Question 2**

Same implementation you need to do perform using HashMap, when you traverse a node you need to save the address of node in hashtable.and in O(1) time you are able to check node is already traverse or not.  
  
  
**Question 3**

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.

