**National University of Computer and Emerging Sciences**



**Laboratory Manual**

*for*

**Data Structures Lab 13**

|  |  |
| --- | --- |
| Course Instructor | Ms. Zareen Alamgir |
| Lab Instructor(s) | Ms. Fariha Maqbool  Ms. Humna Shabir |
| Section | BCS-3F |
| Semester | Fall 2022 |

**Department of Computer Science**

FAST-NU, Lahore, Pakistan

**Objectives:**

In this lab, students will practice:

1. Graphs
2. BFS

**Q1: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 BFS method. One path per line.

Void Print() //to print the BFS tree

};

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);

g.Print();

g.ExploreFunction(1);

g.Print();

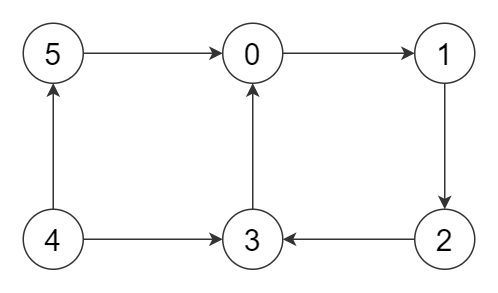
return 0;

}  
  
**Q2:**

A root vertex of a directed graph is a vertex u with a directed path from u to v for every pair of vertices (u, v) in the graph. In other words, all other vertices in the graph can be reached from the root vertex.

A graph can have multiple root vertices. For example, each vertex in a [**strongly connected component**](https://www.techiedelight.com/check-graph-strongly-connected-one-dfs-traversal/) is a root vertex. In such cases, the solution should return anyone of them. If the graph has no root vertices, the solution should return -1.

**Note: Use BFS to perform the task**



**Root vertex: 4**