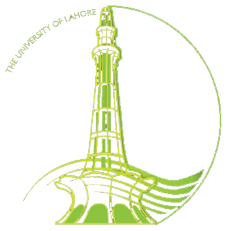

THE UNIVERSITY OF LAHORE
ISLAMABAD CAMPUS

Data Structuers and algorithms (CS09203)

Lab Report

Name:	Irtaza Kashir Raja
Registration #:	SEU-F16-127
Lab Report #:	07
Dated:	21-05-2018
Submitted To:	Mr. Usman Ahmed

The University of Lahore, Islamabad Campus
Department of Computer Science & Information Technology

Experiment # 7
Graphs

Objective
The objective of this session is to understand the various operations on graphs in C++. **Software Tool**
1. Code Blocks with GCC compiler.

1 Theory

Graphs:- Graph is a data structure that consists of following two components: 1. A finite set of vertices also called as nodes. 2. A finite set of ordered pair of the form (u, v) called as edge. The pair is ordered because (u, v) is not same as (v, u) in case of directed graph(di-graph). The pair of form (u, v) indicates that there is an edge from vertex u to vertex v. The edges may contain weight/value/cost.

2 Task

2.1 Task 1

Write a C++ code using functions for the following operations. 1. Add Edges in the adjacent matrix. 2. Display the added Edges.

2.2 Procedure: Task 1

```
#include <iostream>
using namespace std;

class Graph {
private:
      bool** adjMatrix;
      int numVertices;
```

```
public:
      Graph(int numVertices) {
            this->numVertices = numVertices;
            adjMatrix = new bool*[numVertices];
            for (int i = 0; i < numVertices; i++) {
                  adjMatrix[i] = new bool[numVertices];
                  for (int j = 0; j < numVertices; j++)
                        adjMatrix[i][j] = false;
          }
    }

      void addEdge(int i, int j) {
                  adjMatrix[i][j] = true;
                  adjMatrix[j][i] = true;
    }

      void removeEdge(int i, int j) {
                  adjMatrix[i][j] = false;
                  adjMatrix[j][i] = false;
    }

      bool isEdge(int i, int j) {
                  return adjMatrix[i][j];
    }

    void toString() {
      for (int i = 0; i < numVertices; i++) {
                  cout << i << " : ";
                  for (int j = 0; j < numVertices; j++)
                        cout << adjMatrix[i][j] << " ";
                  cout << "\n";
      }

    }

    ~Graph() {
            for (int i = 0; i < numVertices; i++)
                  delete[] adjMatrix[i];
            delete[] adjMatrix;
    }
```

Figure 1: output

```
};


int main(){

 int a,b,c=0,y=0;
        cout<<"how many vertics"<<endl;
         cin>>c;
         Graph g(c);
             while(y!=c){

                        cout<<"add edges "<<endl;
         cin>>a>>b;
         g.addEdge(a,b);

         y++;

             }
         g.toString();


}
```