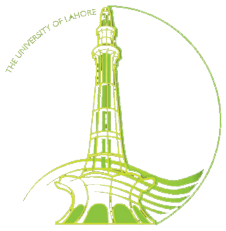

THE UNIVERSITY OF LAHORE
ISLAMABAD CAMPUS

Data Structure And Algorithum

Lab Report

Name:	Ch Hasnain Zafar
Registration #:	SEU-F16-104
Lab Report #:	07
Dated:	16-04-2018
Submitted To:	Mr. Usman Ahmed

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

Experiment # 1
Graph Implementation

Objective
To understand and implement the Graph Problem.

Software Tool
1. DEV C++

1 Theory

The abilities of these data structures are really important for the modern programming. Each of this data structures is used for building a model of real life problems, which are efficiently solved using this model. We will explain what tree data structures are and will review their main advantages and disadvantages. We will present example implementations and problems showing their practical usage. We will focus on binary trees, binary search trees and self-balancing binary search tree.
We will explain what graph is, the types of graphs, how to represent a graph in the memory (graph implementation) and where graphs are used in our life and in the computer technologies. We will see where in.
NET Framework self-balancing binary search trees are implemented and how to use them.

2 Task

2.1 Procedure: Task 1

```
#include<iostream>
#include<iomanip>

using namespace std;
```

```
// A function to print the adjacency matrix.
void PrintMat(int mat[][20], int n)
{
        int i, j;

        cout<<"\n\n"<<setw(4)<<"";
        for(i = 0; i < n; i++)
                cout<<setw(3)<<"("<<i+1<<")";
        cout<<"\n\n";

        // Print 1 if the corresponding vertexes are connected otherwise 0
        for(i = 0; i < n; i++)
        {
                cout<<setw(3)<<"("<<i+1<<")";
                for(j = 0; j < n; j++)
                {
                        cout<<setw(4)<<mat[i][j];
                }
                cout<<"\n\n";
        }
}

int main()
{
        int i, j, v;

        cout<<"Enter the number of vertexes: ";
        cin>>v;

        int  mat[20][20];

        cout<<"\n";
        // Take input of the adjacency of each pair of vertexes.
        for(i = 0; i < v; i++)
        {
                for(j = i; j < v; j++)
                {
                        if(i != j)
                        {
                                cout<<"Enter 1 if the vertex "<<i+1<<" is
```

```
                                        cin>>mat[i][j];

                                        mat[j][i] = mat[i][j];
                                }
                                else
                                        mat[i][j] = 0;
                        }
                }

                PrintMat(mat, v);
}
```