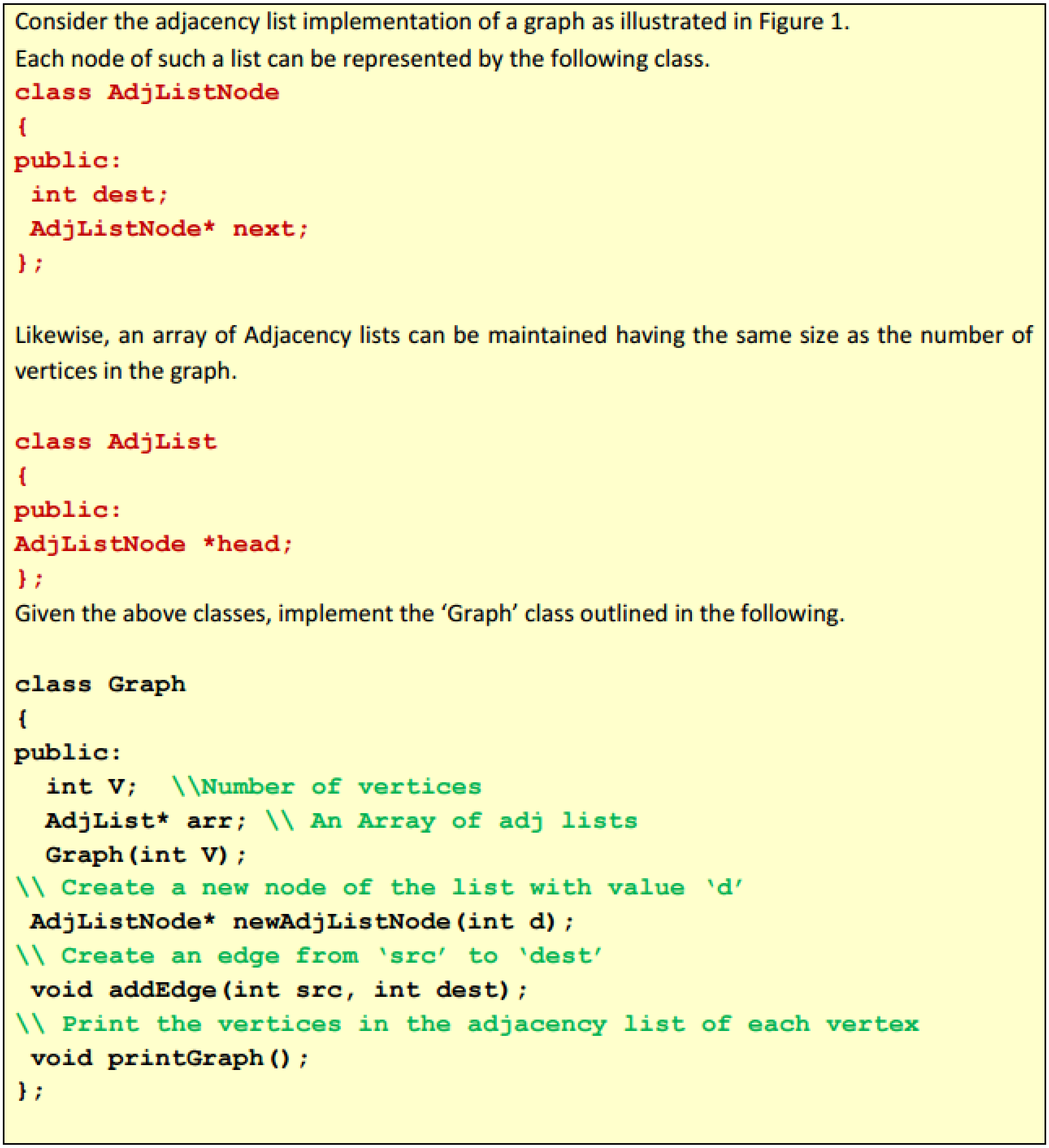
Name: **Muhammad Anas Baig**

Enrollment No.: **01-134152-037**

Section: **BS(CS)-4A**

# LAB-JOURNAL-14

## Exercise 1:



Consider the adjacency list implementation of a graph as illustrated in Figure 1.

Each node of such a list can be represented by the following class.

```
class AdjListNode
{
public:
 int dest;
 AdjListNode* next;
};
```

Likewise, an array of Adjacency lists can be maintained having the same size as the number of vertices in the graph.

```
class AdjList
{
public:
AdjListNode *head;
};
```

Given the above classes, implement the ‘Graph’ class outlined in the following.

```
class Graph
{
public:
  int V;  \\Number of vertices
  AdjList* arr; \\ An Array of adj lists
  Graph(int V);
\\ Create a new node of the list with value ‘d’
 AdjListNode* newAdjListNode(int d);
\\ Create an edge from ‘src’ to ‘dest’
 void addEdge(int src, int dest);
\\ Print the vertices in the adjacency list of each vertex
 void printGraph();
};
```

## Solution:

### node.h File:

```
#pragma once
class node
{
public:
    int dest;
    node *next;
public:
    node(void);
};
```

### node.cpp File:

```
#include "node.h"

node::node(void)
{
}
```

### adjList.h File:

```
#include "node.h"

#pragma once
class adjList
{
public:
    node *head;
public:
    adjList(void);
};
```

### adjList.cpp File:

```
#include "adjList.h"

adjList::adjList(void)
{
}
```

### graph.h File:

```
#include "adjList.h"

#pragma once
class graph
{
public:
    int v;
    adjList *arr;
public:
    graph( int v );
    node * newAdjListNode( int d );
    void addEdge( int src, int dest );
    void printGraph();
};
```

## graph.cpp File:

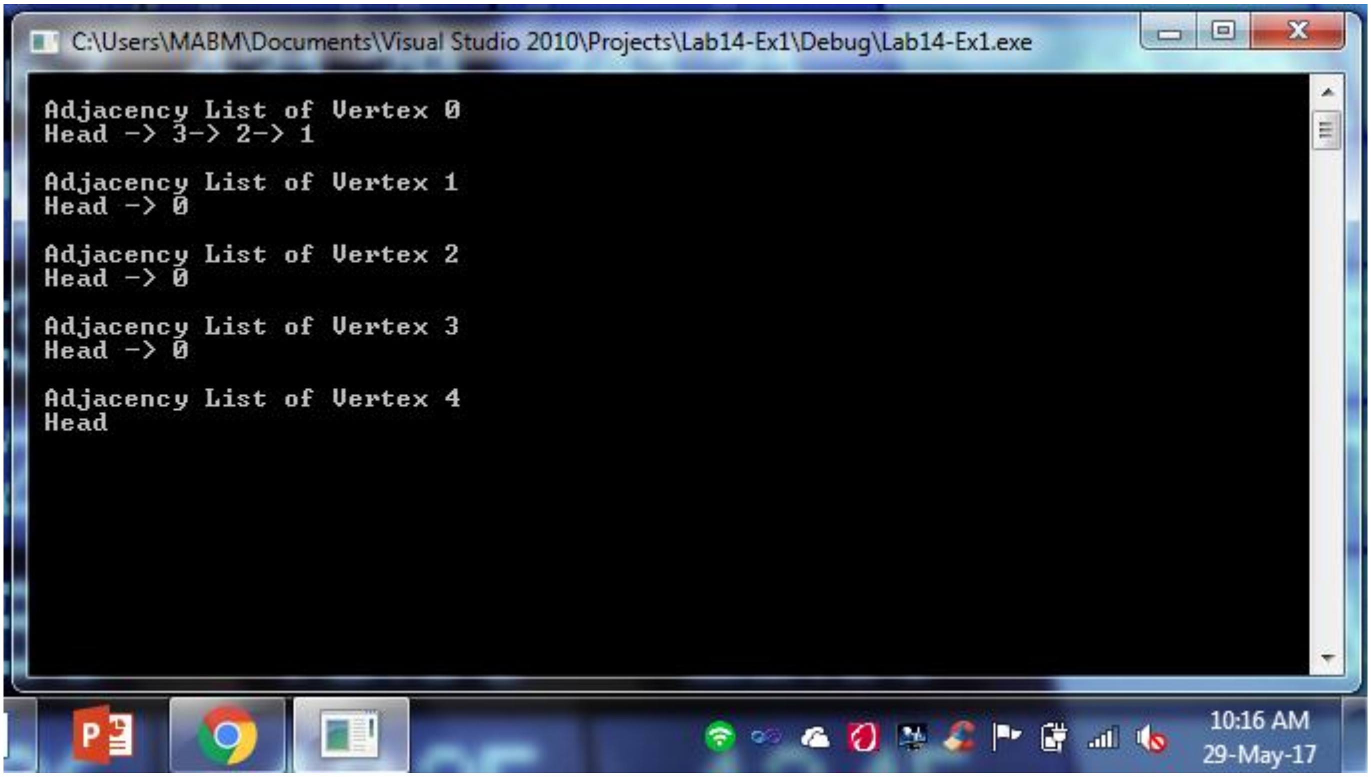
```
#include "graph.h"
#include <iostream>
using namespace std;

graph::graph( int v )
{
    this->v = v;
    arr = new adjList [v];
    for( int i=0; i<v; ++i )
    {
        arr[i].head = NULL;
    }
}

node * graph::newAdjListNode( int dest )
{
    node *ptr = new node;
    ptr->next = NULL;
    ptr->dest = dest;

    return ptr;
}

void graph::addEdge( int src, int dest )
{
    node *ptr = newAdjListNode( dest );
    ptr->next = arr[src].head;
    arr[src].head = ptr;

    ptr = newAdjListNode( src );
    ptr->next = arr[dest].head;
    arr[dest].head = ptr;
}

void graph::printGraph()
{
    for( int V=0; V<v; ++V )
    {
        node * pCrawl = arr[V].head;
        cout<<"\n Adjacency List of Vertex "<<V<<"\n Head ";
        while( pCrawl )
        {
            cout<<"-> "<<pCrawl->dest;
            pCrawl = pCrawl->next;
        }
        cout<<endl;
    }
}
```

## main.cpp File:

```
#include"adjList.h"
#include"graph.h"
#include"node.h"
#include"conio.h"
#include<iostream>
using namespace std;

```

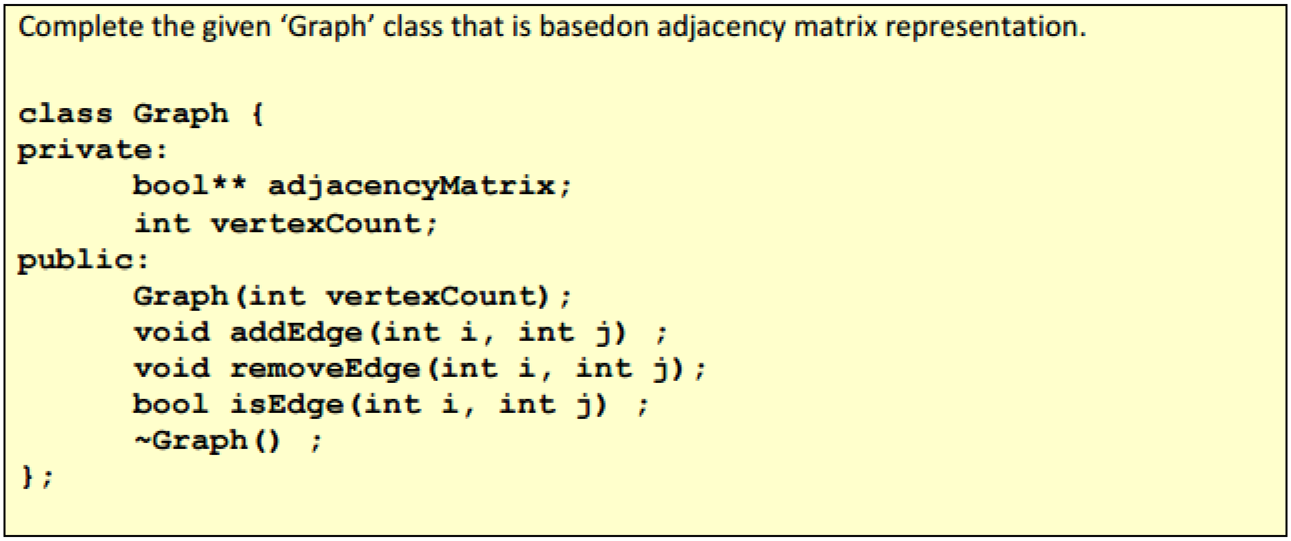
```
int main()
{
    graph g(5);

    g.addEdge( 0, 1 );
    g.addEdge( 0, 2 );
    g.addEdge( 0, 3 );
    //now VERTEX 0 has 3 Edges towards  VERTEX 1, 2, 3 and also each VERTEX 1, 2, 3 has Edge towards VERTEX 0
    g.printGraph();

    getch();
}
```

## Output:


C:\Users\MABM\Documents\Visual Studio 2010\Projects\Lab14-Ex1\Debug\Lab14-Ex1.exe
10:16 AM
29-May-17

```
Adjacency List of Vertex 0
Head -> 3-> 2-> 1

Adjacency List of Vertex 1
Head -> 0

Adjacency List of Vertex 2
Head -> 0

Adjacency List of Vertex 3
Head -> 0

Adjacency List of Vertex 4
Head
```

## Exercise 2:



Complete the given ‘Graph’ class that is basedon adjacency matrix representation.

```
class Graph {
private:
      bool** adjacencyMatrix;
      int vertexCount;
public:
      Graph(int vertexCount);
      void addEdge(int i, int j)  ;
      void removeEdge(int i, int j);
      bool isEdge(int i, int j)  ;
      ~Graph()  ;
};
```

## Solution:

### node.h File:

```
#pragma once
class node
{
public:
    int dest;
    node *next;
public:
    node(void);
};
```

### node.cpp File:

```
#include "node.h"

node::node(void)
{
}
```

### graph.h File:

```
#include "adjList.h"

#pragma once
class graph
{
private:
    bool** adjacencyMatrix;
      int vertexCount;
public:
    graph( int vertexCount );
    void addEdge( int i, int j ) ;
    void removeEdge( int i, int j );
    bool isEdge( int i, int j ) ;
    ~graph();
};
```

## graph.cpp File:

```
#include "graph.h"
#include <iostream>
using namespace std;

graph::graph( int vertexCount )
{
    this->vertexCount = vertexCount;
    adjacencyMatrix = new bool*[vertexCount];
    for( int i=0; i<vertexCount; i++)
    {
        adjacencyMatrix[i] = new bool[vertexCount];
        for( int j=0; j<vertexCount; j++)
        {
            adjacencyMatrix[i][j] = false;
        }
    }
}

void graph::addEdge(int i, int j)
{
    if (i >= 0 && i < vertexCount && j > 0 && j < vertexCount )
    {
        adjacencyMatrix[i][j] = true;
        adjacencyMatrix[j][i] = true;
    }
}

void graph::removeEdge(int i, int j)
{
    if( i >= 0 && i < vertexCount && j > 0 && j < vertexCount )
    {
        adjacencyMatrix[i][j] = false;
        adjacencyMatrix[j][i] = false;
    }
}

bool graph::isEdge(int i, int j)
{
    if( i >= 0 && i < vertexCount && j > 0 && j < vertexCount )
    {
        return adjacencyMatrix[i][j];
    }
    else
    {
        return false;
    }
}
graph::~graph()
{
    for( int i=0; i<vertexCount; i++)
    {
        delete[] adjacencyMatrix[i];
        delete[] adjacencyMatrix;
    }
}
```

## adjList.h File:

```
#include "node.h"

#pragma once
class adjList
{
public:
    node *head;
public:
    adjList(void);
};
```

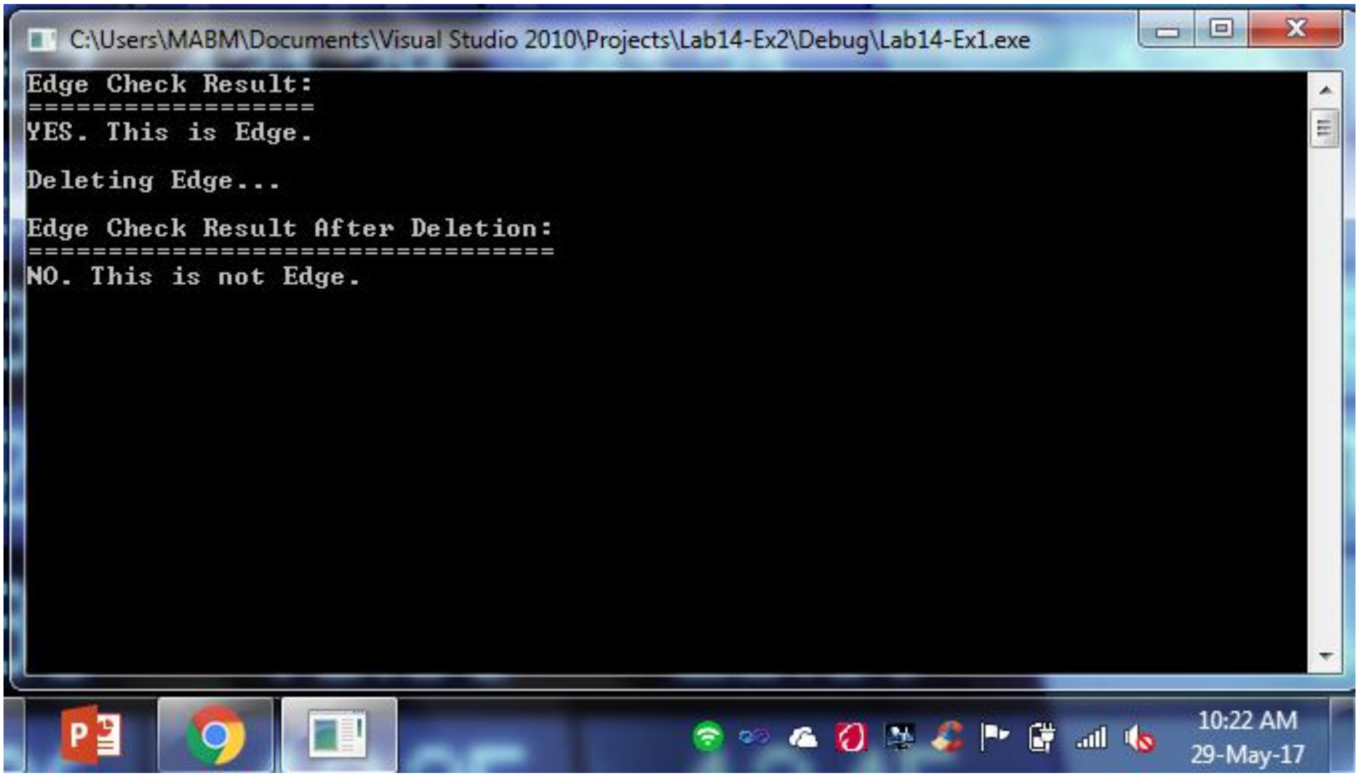
## adjList.cpp File:

```
#include "adjList.h"

adjList::adjList(void)
{
}
```

## main.cpp File:

```
#include"adjList.h"
#include"graph.h"
#include"node.h"
#include"conio.h"
#include<iostream>
using namespace std;

int main()
{
    graph g(5);

    g.addEdge( 0, 1 );
    g.addEdge( 0, 2 );
    g.addEdge( 0, 3 );
    //now VERTEX 0 has 3 Edges towards  VERTEX 1, 2, 3 and also each VERTEX 1, 2, 3 has Edge towards VERTEX 0

    cout<<"Edge Check Result:"<<endl;
    cout<<"=================="<<endl;
    if( g.isEdge( 0, 2 ) )
    {
        cout<<"YES. This is Edge."<<endl;
    }
    else
    {
        cout<<"NO. This is not Edge."<<endl;
    }

    cout<<endl<<"Deleting Edge..."<<endl;
    g.removeEdge( 0, 2 );
    cout<<endl;

    cout<<"Edge Check Result After Deletion:"<<endl;
    cout<<"================================="<<endl;
    if( g.isEdge( 0, 2 ) )
    {
        cout<<"YES. This is Edge."<<endl;
    }
    else
    {
        cout<<"NO. This is not Edge."<<endl;
    }

    getch();
}
```

## Output:



C:\Users\MABM\Documents\Visual Studio 2010\Projects\Lab14-Ex2\Debug\Lab14-Ex1.exe
10:22 AM
29-May-17

```
Edge Check Result:
==================
YES. This is Edge.

Deleting Edge...

Edge Check Result After Deletion:
=================================
NO. This is not Edge.
```