**Lab 09**

**Name:** Will Townsend

**Class:** COSC 320 – Section 751

**Date:** 11/19/2020

**Lab Report:** This lab was fairly simple to execute. The most confusing thing to do was putting the file together and making it work properly. There is a small error in regard to the input of the graph where it copies the values in the last line twice, but everything else works fine. It took me about an hour total to complete. This lab was done 100% independently without help from outside resources (i.e. classmates or internet).

**Prelab:**

graphA.dat:

A B 0

A D 0

B C 0

C A 0

C F 0

D E 0

E A 0

F C 0

**Lab Task:**

Task 2:

#include<iostream>

#include<fstream>

#include<stdio.h>

#include<set>

#include<utility>

#include"d\_graph.h"

int main(){

graph<char> graphA;

graphA.insertVertex('A');

graphA.insertVertex('B');

graphA.insertVertex('C');

graphA.insertVertex('D');

graphA.insertVertex('E');

graphA.insertVertex('F');

std::ifstream fileA;

fileA.open("graphA.dat",std::fstream::in);

puts("Lab 09:");

while(fileA){

char v1,v2;

int w;

fileA>>v1;

fileA>>v2;

fileA>>w;

graphA.insertEdge(v1,v2,w);

}

fileA.close();

graphA.insertEdge('F','D',1);//i

graphA.eraseVertex('B');//ii

graphA.eraseEdge('A','D');//iii

//iv

char input;

printf("Enter a Vertex(A,C,D,E,F): ");

std::cin>>input;

printf("\nNeighbor(s) of %c:\n",input);

set<char> neighbor=graphA.getNeighbors(input);

std::set<char>::iterator curr=neighbor.begin();

if(neighbor.empty())

std::cout<<"None";

else{

for(int i=0;i<neighbor.size();i++){

std::cout<<\*(curr++);

if (i!=neighbor.size()-1)

std::cout<<" & ";

}

}

puts("\n");

graphA.insertVertex('G');//v

//vi

graphA.insertEdge('G','C',1);

graphA.insertEdge('G','F',1);

graphA.insertEdge('D','G',1);

//vii

std::cout<<"Graph A:"<<std::endl<<graphA;

return 0;

}

**Sample Outputs:**

Output 1:

Lab 09:

Enter a Vertex(A,C,D,E,F): A

Neighbor(s) of A:

None

Graph A:

A: in-degree 2 out-degree 0

Edges:

G: in-degree 1 out-degree 2

Edges: C (1) F (1)

C: in-degree 2 out-degree 2

Edges: A (0) F (0)

D: in-degree 1 out-degree 2

Edges: G (1) E (0)

E: in-degree 1 out-degree 1

Edges: A (0)

F: in-degree 2 out-degree 2

Edges: C (0) D (1)

Output 2:

Lab 09:

Enter a Vertex(A,C,D,E,F): C

Neighbor(s) of C:

A & F

Graph A:

A: in-degree 2 out-degree 0

Edges:

G: in-degree 1 out-degree 2

Edges: C (1) F (1)

C: in-degree 2 out-degree 2

Edges: A (0) F (0)

D: in-degree 1 out-degree 2

Edges: G (1) E (0)

E: in-degree 1 out-degree 1

Edges: A (0)

F: in-degree 2 out-degree 2

Edges: C (0) D (1)

Output 3:

Lab 09:

Enter a Vertex(A,C,D,E,F): D

Neighbor(s) of D:

E

Graph A:

A: in-degree 2 out-degree 0

Edges:

G: in-degree 1 out-degree 2

Edges: C (1) F (1)

C: in-degree 2 out-degree 2

Edges: A (0) F (0)

D: in-degree 1 out-degree 2

Edges: G (1) E (0)

E: in-degree 1 out-degree 1

Edges: A (0)

F: in-degree 2 out-degree 2

Edges: C (0) D (1)

Output 4:

Lab 09:

Enter a Vertex(A,C,D,E,F): E

Neighbor(s) of E:

A

Graph A:

A: in-degree 2 out-degree 0

Edges:

G: in-degree 1 out-degree 2

Edges: C (1) F (1)

C: in-degree 2 out-degree 2

Edges: A (0) F (0)

D: in-degree 1 out-degree 2

Edges: G (1) E (0)

E: in-degree 1 out-degree 1

Edges: A (0)

F: in-degree 2 out-degree 2

Edges: C (0) D (1)

Output 5:

Lab 09:

Enter a Vertex(A,C,D,E,F): F

Neighbor(s) of F:

C & D

Graph A:

A: in-degree 2 out-degree 0

Edges:

G: in-degree 1 out-degree 2

Edges: C (1) F (1)

C: in-degree 2 out-degree 2

Edges: A (0) F (0)

D: in-degree 1 out-degree 2

Edges: G (1) E (0)

E: in-degree 1 out-degree 1

Edges: A (0)

F: in-degree 2 out-degree 2

Edges: C (0) D (1)