# Department of Computing

# School of Electrical Engineering and Computer Science

**CS-250: Data Structure and Algorithms**

**Class: BSCS 10C**

**Lab 11:  Graph Fundamentals**

# Name: Saad Subhani, Moez Ahmed

# Section: BSCS-10C

# CMS: 338803,354647

**Date: 10th December, 2021**

**Time: 9:00 am – 11:50 am**

# Instructor: Prof. Dr. Faisal Shafait

# Lab Engineer: Mr. Aftab Farooq

**Lab 11: Graph Fundamentals**

**Introduction**

In this lab, you will implement Graphs and get familiarize to them.

**Objectives**

Objective of this lab is to implement Graphs and understand their fundamentals.

**Tools/Software Requirement**

Visual Studio C++

**Description**

Consider a subset of the social network of the class with only those students whose names start with the letter A. We assume all students are friends of each other on the network.

Each student makes some posts which are liked by some of the students. We track whose post is liked by whom and record that data in a matrix.

Each row header of the matrix represents a person creating the post.

Each column header of the matrix represents a person liking the post.

Considering a simple scenario of three students A, B, C, the matrix looks like this

A B C

A 4

B 1

C 2

The data above represents

- A liked 1 post of B and 2 posts of C

- B liked 4 posts of A

- C did not like any post

**Task:**

Use the code given in the tutorial

<http://theoryofprogramming.com/2014/12/24/graph-theory-basics/>

to represent the social network given in the file social-network.xlsx

Modify the code so that you only print the names of popular students (i.e. those students whose posts received more than 10 likes)

**Important Note:** Practice your knowledge of OOP with C++ when creating a solution.

**Lab Grading:**

|  |  |
| --- | --- |
| **Task** | **Marks** |
| Lab Viva/Quiz | 5 |
| Comments/ Indentation | 2 |
| Solution Document | 2 |
| Output Screen Shots | 1 |
| -- | -- |
| Total | 10 |

# Code:

#include <cstdio>

#include <vector>

#include <list>

#include <utility>

#include <iostream>

#include <string>

#include <fstream>

#include <sstream>

using namespace std;

vector<vector<int>> intData; *//public vector to read likes*

vector<string> names; *//public vector to read names*

void read\_record()

{

*// File pointer*

    fstream fin;

*// Open an existing file*

    fin.open("social\_network.csv");

    if (!fin)

    {

        cout << "Could not open file" << endl;

        return;

    }

*// Read the Data from the file*

*// as String Vector*

    vector<string> stringRow;

    vector<int> intRow;

    string line, word, temp;

    bool flag = false;

    while (fin >> temp)

    {

        stringRow.clear();

        intRow.clear();

*// read an entire row and*

*// store it in a string variable 'line'*

        getline(fin, line);

*// used for breaking words*

        stringstream lineStream(line);

*// read every column data of a row and*

*// store it in a string variable, 'word'*

        if (flag) *//flag has been set so that the first row containing names strings is not pushed to the vector*

        {

            while (getline(lineStream, word, ','))

            {

*// add all the column data*

*// of a row to a vector*

                stringRow.push\_back(word);

            }

            for (int i = 1; i < stringRow.size(); i++)

            {

                intRow.push\_back(stoi(stringRow[i]));

            }

            intData.push\_back(intRow);

        }

        flag = true;

    }

}

void readNames()

{ *//reading all the names from the CSV file*

    fstream fin;

    fin.open("social\_network.csv");

    if (!fin)

    {

        cout << "Could not open file" << endl;

        return;

    }

    string line, word;

    getline(fin, line);

    int count = 1;

*// used for breaking words*

    stringstream lineStream(line);

    while (getline(lineStream, word, ',') && count < 10) *//count < 10 to read only the names that Start with A*

    {

*// add all the column data*

*// of a row to a vector*

        if (count != 1)

        {

            names.push\_back(word);

        }

        count++;

    }

}

int main()

{

    read\_record(); *//got the likes from the CSV file*

    readNames(); *//got all the names from the CSV file*

    int edges, v1, v2, weight;

    int popularity[50] = {0};

*// Adjacency List is a vector of list.*

*// Where each element is a pair<int, int>*

*// pair.first -> edge's destination*

*// pair.second -> edge's weight*

    vector<list<pair<int, int>>> adjacencyList(names.size() + 1);

    for (int i = 1; i <= names.size(); ++i)

    {

        for (int j = 1; j <= intData.size(); j++)

        {

            if (intData[i - 1][j - 1] != 0) *//only non zero values are being read*

            { *// Adding Edges to the Graph*

                v1 = i; *//first argument is v1*

                v2 = j; *//secng argument is the person where like is coming from v2*

                weight = intData[i - 1][j - 1]; *//value is the number of likes i.e.e weight*

                adjacencyList[v1].push\_back(make\_pair(v2, weight));

*//  adjacencyList[v2].push\_back(make\_pair(v1, weight));*

            }

        }

    }

    printf("\nThe Adjacency List-\n");

*// Printing Adjacency List*

    for (int i = 1; i < adjacencyList.size(); ++i)

    {

        cout << names[i - 1] << endl;

        list<pair<int, int>>::iterator itr = adjacencyList[i].begin();

        while (itr != adjacencyList[i].end())

        {

            printf(" -> %d(%d)", (\*itr).first, (\*itr).second);

            popularity[i] = popularity[i] + (\*itr).second;

            ++itr;

        }

        printf("\n");

    }

    cout << "\nThe list of popular students is: \n"

         << endl;

    for (int i = 1; i <= names.size(); i++)

    {

        if (popularity[i] > 10) *//the students having likes greater than 10*

        {

            cout << names[i - 1] << " having " << popularity[i] << " likes" << endl;

        }

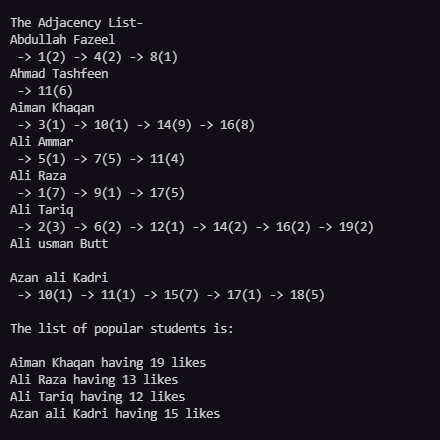
    }

    cout << endl;

    return 0;

}

# Output:



**Deliverables**

This lab grading policy is as follows: The lab is graded between 0 to 10 marks. The submitted solution can get a maximum of 5 marks. At the end of each lab or in the next lab, there will be a viva related to the tasks. The viva has a weightage of 5 marks. Insert the solution/answer in this document. You must show the implementation of the tasks in the designing tool, along with your complete Word document to get your work graded. You must also submit this Word document on the LMS. In case of any problems discuss it by emailing it to [aftab.farooq@seecs.edu.pk](mailto:aftab.farooq@seecs.edu.pk).

**Note:** Students are required to upload the lab on LMS before deadline.

Use proper indentation and comments. Lack of comments and indentation will result in deduction of marks.

The End