CS 202 Fundamental Structures of Computer Science II
Assignment 5 – Graphs

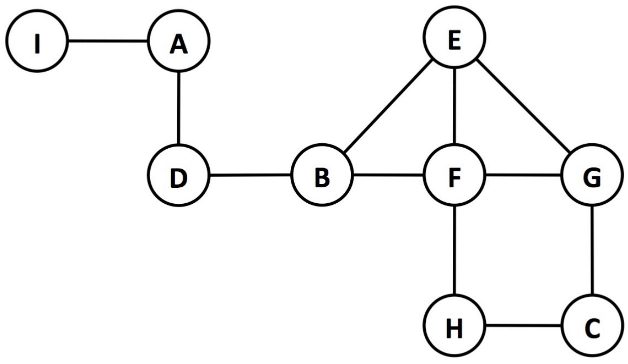
Assigned on: 27 April 2016 (Wednesday)
Due Date: 9 May 2016 (Monday)

Question-1 (20 points)

For the graph given below, give the sequence of vertices when they are traversed starting from *vertex A* using

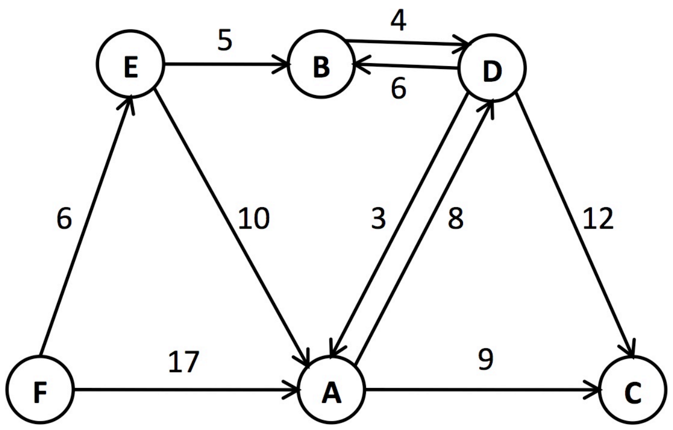
- The depth first traversal algorithm
- The breadth first traversal algorithm

In your solution, for a vertex, use the alphabetical order to visit its adjacent vertices.



Question-2 (10 points)

For the following weighted directed graph, find the shortest paths from *vertex F* to all other vertices using the Dijkstra's shortest path algorithm. Show all steps of the Dijkstra's algorithm.



Question-3 (70 points)

Programming Assignment: In this question, you are asked to implement a simple friendship network. In this implementation, you will represent this friendship network by a graph and answer the queries, each of which corresponds to calling a member function, on this graph.

Below is the required public part of the **`FriendNet`** class that you will implement in this assignment. The name of the class must be **`FriendNet`**, and must include these public member functions. We will use these functions to test your code. The interface for the class must be written in a file called **`FriendNet.h`** and its implementation must be written in a file called **`FriendNet.cpp`**. You can define additional public and private member functions and private data members in this class. You can also define additional classes in your solution.

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

class FriendNet{

public:

     FriendNet(const string fname);
     FriendNet(const FriendNet& aNet);       // copy contructor
     ~FriendNet();                           // destructor

     void listFriends(const string pname);
     void listPeople(const string pname, const int hno);

     void displayAverageDegrees();
     void displayAverageClusteringCoefficients();
     void displayDiameters();


     // ...
     // define other public member functions here,
     // if you have any


private:

     // ...
     // define your data members here
     // define private member functions here, if you have any

};
```

The details of the member functions are as follows:

- **`FriendNet(const string fname);`**

  The default constructor loads a friendship network from an input file called **`fname`**. The first row of this file indicates the number of people in the network and each subsequent row includes information of a particular person. This information contains <id> <name> <degree> <friend id>* tokens separated by white space. For a particular person *P*,

  - **<id>** and **<name>** are the id and the name of person *P*, respectively.
  - **<degree>** is the number of friends of person *P*.
  - **<friend id>** is the id of a friend of person *P*. Note that a person may have zero or more friends and there will be exactly **<degree>** friend ids after the degree token.

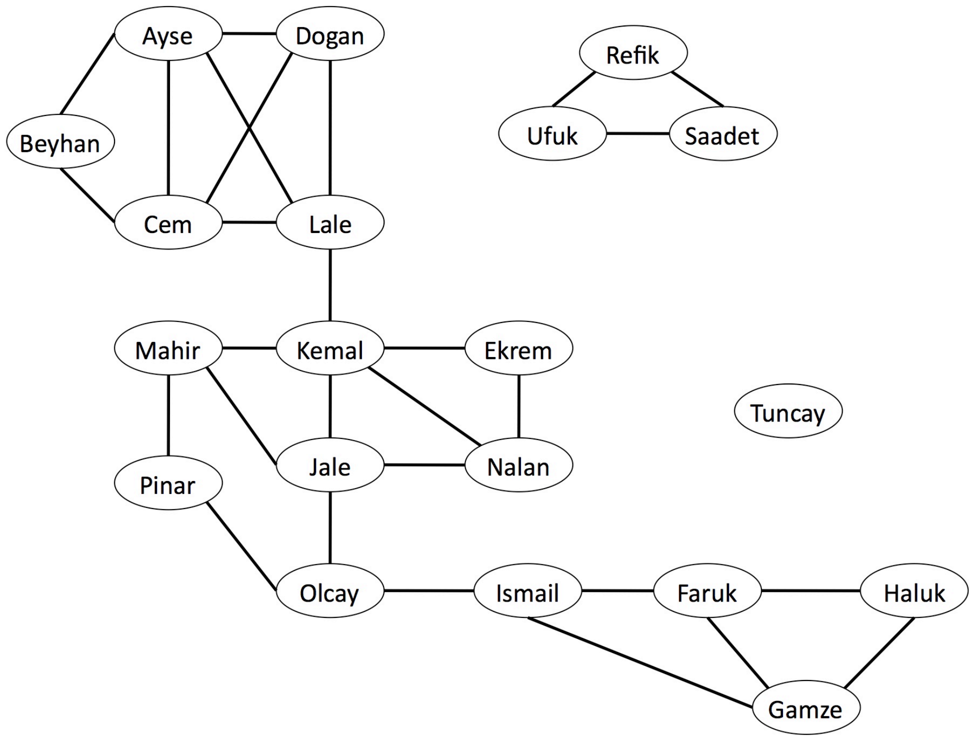
  In this assignment, you may assume that the contents of the input file are always valid. You may also assume that the ids and the names are unique and the ids are in the range of 0 and N – 1, where N is the number of people in the network. You may also assume that all names are case sensitive.

  Note that, if the input file **`fname`** does not exist, then the default constructor creates an empty friendship network.

  The following table shows an example input file for the network illustrated below. This file contains 20 people, as indicated in its first line. For example, the sixth line of this file indicates that the person with an id of 4 has the name of Dogan and has three friends, with the ids of 0, 2, and 3.

  The input file of the network illustrated below:

```
20
0  Ayse        4 1 4 3 2
1  Beyhan      2 0 2
2  Cem         4 0 4 1 3
3  Lale        4 18 0 4 2
4  Dogan       3 0 2 3
5  Ekrem       2 18 9
6  Refik       2 7 19
7  Saadet      2 6 19
8  Tuncay      0
9  Nalan       3 10 18 5
10 Jale        4 16 18 9 14
11 Gamze       3 12 15 13
12 Haluk       2 11 15
13 Ismail      3 14 15 11
14 Olcay       3 10 17 13
15 Faruk       3 13 11 12
16 Mahir       3 10 17 18
17 Pinar       2 16 14
18 Kemal       5 3 10 16 9 5
19 Ufuk        2 6 7
```



- **`void listFriends(const string pname);`**

  It lists the names of the friends of a given person whose name is **`pname`**. If this given person does not take place in the friendship network, give a warning message.

  See the output example below for the format. You may assume that the names are unique within the friendship network.

- **`void listPeople(const string pname, const int hno);`**

  It lists the names of all people that are accessible from a given person, whose name is **`pname`**, within the given number **`hno`** of hops. If this given person does not take place in the friendship network, give a warning message. If the given number of hops is non-positive, this function should not list any people.

  See the output example below for the format. Similarly, you may assume that the names are unique within the friendship network.

- **`void displayAverageDegrees();`**

  It calculates and displays the average degree of each connected component within the friendship network. The degree of a vertex is defined as the number of its neighbors. The average degree of a connected component is the mean of the degrees computed for every vertex in this connected component.

- **`void displayAverageClusteringCoefficients();`**

  It calculates and displays the average clustering coefficient of each connected component within the network. The clustering coefficient of a vertex quantifies the connectivity information in its neighborhood. The clustering coefficient *CC*(*v*) of *vertex v* is defined as

$$CC(v) = \frac{2 \cdot E(v)}{d(v) \cdot [d(v) - 1]}$$

where *d*(*v*) is the number of neighbors of *vertex v* and *E*(*v*) is the number of existing edges between these neighbors. The clustering coefficient of a vertex with 0 or 1 neighbor is 0. The average clustering coefficient of a connected component is the mean of the clustering coefficients computed for every vertex in this connected component.

- **`void displayDiameters();`**

  It calculates and displays the diameter of each connected component within the friendship network. The diameter of a connected component is the longest of the shortest paths between any pair of vertices within this connected component.

Below is an example test program that uses this class and the corresponding output. This test program uses the friendship network illustrated above. Assume that the name of the input file is “friends”. We will use a similar program to test your solution; so make sure that the name of the class is **`FriendNet`**, its interface is in the file called **`FriendNet.h`**, and the required member functions are defined as shown above. (Of course, we will use other test cases to test your program.)

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

int main(){
    FriendNet F("friends");

    F.listFriends("Ayse");
    F.listFriends("Tuncay");
    F.listFriends("Cigdem");
    cout << endl;

    F.listPeople("Cem",2);
    F.listPeople("Ufuk",3);
    F.listPeople("Tuncay",2);
    F.listPeople("Cigdem",4);
    F.listPeople("Demir",-3);
    F.listPeople("Cem",-1);
    cout << endl;

    F.displayAverageDegrees();
    cout << endl;

    F.displayAverageClusteringCoefficients();
    cout << endl;

    F.displayDiameters();

    return 0;
}
```

The output of this program will be as follows.

```
Ayse has 4 friends: Beyhan, Dogan, Lale, Cem
Tuncay does not have any friends.
Cigdem does not exist in the network.

People accessible from Cem within 2 hops: Ayse, Dogan, Beyhan, Lale, Kemal
People accessible from Ufuk within 3 hops: Refik, Saadet
Cannot access to any people from Tuncay
Cigdem does not exist in the network.
Demir does not exist in the network.
-1 is not a valid hop number.

There are 3 connected components. The average degrees are:
For component 1: 3.125
For component 2: 2.000
For component 3: 0.000

There are 3 connected components. The average clustering coefficients are:
For component 1: 0.571
For component 2: 1.000
For component 3: 0.000

There are 3 connected components. The diameters are:
For component 1: 8
For component 2: 1
For component 3: 0
```

HAND-IN

- Before 23:59 of May 9, 2016, upload your solutions using the following online submission form, http://www.cs.bilkent.edu.tr/~saksoy/courses/cs202-Spring2016/upload.html. You should upload a single zip file that contains
  - Your answers to questions 1 and 2 in a file named firstname_lastname_hw5.pdf. Do NOT send photos of your solution; type your answers.
  - Your code files for question 3. The names of your code files must be FriendNet.h, and FriendNet.cpp. You can also write additional files but do not include your test main function in any of these files.
  - Do not put any unnecessary files such as the auxiliary files generated from your favorite IDE. Be careful to avoid using any OS dependent utilities.
  - Do not forget to write your id, name, section, assignment number and any other information relevant to your program in the beginning of every file that you are submitting.
  - Do not forget to write comments at important parts of your code.
  - Please put all the necessary files in a folder named firstname_lastname_hw5 and then compress this folder in a single zip file.
  - **Any violation of these may cause a significant loss from your grade.**

- For this assignment, you must use your own implementations of graphs. You can adapt the codes and pseudocodes in the Carrano book. However, you are NOT allowed to use any existing codes from other sources (including the codes given in other textbooks, found on the internet, and belonging to your classmates). Furthermore, you are NOT allowed to use any data structure or function from the C++ standard template library (STL).
- Although you may use any platform or any operating system to implement your solutions, your code should work in a Linux environment (specifically using the g++ compiler). We will compile your programs with the g++ compiler and test your codes in a Linux environment. Thus, you may lose a significant amount of points, if your C++ code does not compile or execute in a Linux environment.
- Keep all the files before you receive your grade.
- This homework will be graded by your TA Nabil AbuBaker (nabil.abubaker at bilkent edu tr). Thus, you may ask your homework related questions directly to him.

DO THE HOMEWORK YOURSELF. PLAGIARISM AND CHEATING ARE HEAVILY PUNISHED!!!