Computing Structures – Fall 2022
Project 2
Due: September 30th 2022 at 11:59 PM

Objective:
This project builds upon the previous project 1. You will learn about template, exceptions handling, de allocating memory, overloading operators and destructing objects.

Addendum:
- The Node class and the Edge class now have more information to be stored. The corresponding information is also given in the input file. (below is an explanation of the input file and the new commands).
- The Node class and the edge class is now templated – refer to the boilerplate code provided for project 2.
- Deletion method in graphDB: If numEdges becomes half of maxEdges, you re allocate memory for myEdges to make it smaller (half of current maxEdges).
- Implement ostream operator for the GraphDB class – this will be instead of the display method already written. You need to call this from the main function to display the nodes and the edges of masterGraph (for option D).
- Implement exception handling for when there is an edge that is trying to add a non-existent node. Refer to the book for an example exception class.
- Implement the findNeighbours method in the GraphDB class – given a nodeNumber, return an array of neighbours of the given node.
- Implement a destructor method for all the three classes making sure you destruct all memory allocated.

Input explanation:
The input file given is the in the following format:
```
4
4

0 Jack 01051995 Norman
1 Tom 02051996 Pawhuska
2 Anna 03111995 Ada
3 Beth 04021997 Clinton

I 0 1 friend 2
I 0 2 boss 3
I 2 3 friend 5
R 0 1
N 0
D
E 0 2
…
```

The input file starts with the number of nodes (4) that are given followed by the maximum number of edges (4). Then we list all the nodes that consists of the node number, name, year created and location.
After this, we give you a list of commands/options that go till the end of the input file. The commands can either be I, R, D, N or E:

- I – the insert edge command is followed by the two nodes numbers it is connecting along with the edgeInfo and the years known.
- R – the remove edge command is followed by the edge we want to remove represented by the two nodes the edge is connecting.
- D – the display command displays the nodes and edges that are currently in the data structure in the specified format (sample output file).
- E – this operation asks if there exists an edge between the given two nodes.
- N – this operation followed by a nodeNum (x) returns an array of nodeNums that are the neighbours of node x.

Class definition:

A boiler plate source file has been provided along with this project specification. The class definition and a main function have also been given with comments to get you started with the project.

A sample output file will be provided (in the next few days) for the given corresponding input file. Your output is supposed to exactly be the same as this output. You need to essentially follow this format of the output. This is necessary for your program to pass the autograder.

Redirected input:

Redirected input provides you a way to send a file to the standard input of a program without typing it using the keyboard. To use redirected input in Visual Studio environment (on windows), follow these steps: After you have opened or created a new project, on the menu go to project, project properties, expand configuration properties until you see Debugging, on the right you will see a set of options, and in the command arguments type <"input filename". The < sign is for redirected input and the input filename is the name of the input file (including the path if not in the working directory).

If you use macOS or linux (or windows using powershell), you may use the terminal to compile and run your program using g++. This is the compiler that we use in the autograder on GradeScope (you may need to install g++ on your computer). Once you installed g++, you may compile your program using the following command:

```
g++ project2.cpp -o p2
```

You may then run the program with the redirected input using the following command:

```
./p2 < input1.txt
```

Make sure your program and your input file are in the same folder.

A simple program that reads the input file and displays everything that was read is given below.

```
#include <iostream>

using namespace std;

int main ()
{
	int numNodes, maxEdges;

	cin >> numNodes >> maxEdges;
	cout << numNodes << maxEdges;

	return 0;
}
```

Submission:

Submission will be through GradeScope. Your program will be autograded with test cases and then hand graded to check for documentation and if you have followed the specifications given. You may submit how many ever times to check if your program passes the test cases provided. Test cases will be released at the beginning and later other test cases will be released while grading. For the autograder to work, the program you upload must be named as **project2.cpp**.

Your final submission must contain your source file named **project2.cpp**. You access GradeScope using the tab on the left in our course page on canvas.

Sample output file for corresponding input files will be released. The input1.txt file given is a simple input file that will help you understand the project, more complicated ones will be released later and used for grading.

Constraints:

1. In this project, the only header you will use the header files that are given to you in the boiler plate code. Using other or excess header files will be subject to a heavy grade penalty for the project.
2. None of the projects is a group project. Consulting with other members of this class on programming projects is strictly not allowed and plagiarism charges will be imposed on students who do not follow this.
    - Please review academic integrity policies in the modules.

How to ask for help:

1. Check FAQ discussion board of the project first.
2. Email the TA with your precise questions.
3. Upload your well commented program to CodePost.
    a. This is a website which is used to share code.
    b. You will upload your program and I can view it and comment on it.

c. Here is the invite link to our class for the summer.
d. Once you join the class on CodePost, you can upload your program to the Project 1 assignment.

   Note: Your program will not be auto graded at CodePost, this is just for when you want to ask a question and a place where I can look at your program and comment on it. CodePost is great for live feedback. GradeScope is the place where you should submit and where your program will be autograded.

End of project 2 description.

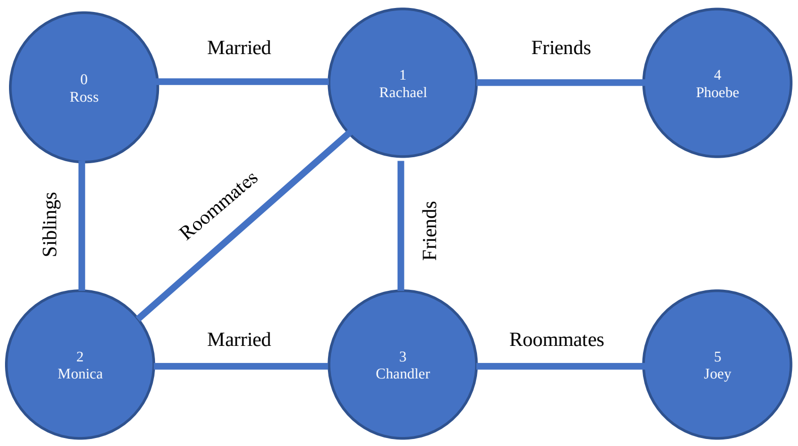
Project 1 description:

Description:

A graph data structure consists of a set of nodes connected using a set of edges. Mathematically represented in graph theory as G = (V, E) where V is the vertex/node set and E is the edge set. One edge connects exactly two nodes that are in the graph.

A Graph Database is a graph in which we can store information and relations and query information as needed. They are most popularly used to store the information on social networks where a person may be represented as a node and an edge may represent the friendship/connection of one node to another.

Below is an example of a graph where each node consists of a node number and a string/name as well. The edges that are connecting these nodes also have a string attached to each of them. The string in the edge represents the relationship between the two nodes that are being connected.



In this project, you will create the graph structure for storing this type of information read from the input file. You are given a set of nodes that are constant, and edges maybe added or deleted. The maximum number of edges is given to you that may have to be resized if the number of edges goes over the given maximum number of edges.

You are required to write all the methods that are given in the boilerplate code.

Input explanation:

The input file given is the in the following format:

```
4
4

0 Jack
1 Tom
2 Anna
3 Beth

I 0 1 friend
I 0 2 boss
I 2 3 friend
R 0 1
D
E 0 2
…
```

The input file starts with the number of nodes (4) that are given followed by the maximum number of edges (4). Then we list all the nodes that consists of the node number and the corresponding string.

After this, we give you a list of commands/options that go till the end of the input file. The commands can either be I, R, D or E:

- I – the insert edge command is followed by the two nodes numbers it is connecting along with the edgeInfo.
- R – the remove edge command is followed by the edge we want to remove represented by the two nodes the edge is connecting.
- D – the display command displays the nodes and edges that are currently in the data structure in the specified format (sample output file).
- E – this operation asks if there exists an edge between the given two nodes.

Class definition:

A boiler plate source file has been provided along with this project specification. The class definition and a main function have also been given with comments to get you started with the project.

A sample output file will be provided (in the next few days) for the given corresponding input file. Your output is supposed to exactly be the same as this output. You need to essentially follow this format of the output. This is necessary for your program to pass the autograder.

Redirected input:

Redirected input provides you a way to send a file to the standard input of a program without typing it using the keyboard. To use redirected input in Visual Studio environment (on windows), follow these steps: After you have opened or created a new project, on the menu go to project, project properties, expand configuration properties until you see Debugging, on the right you will see a set of options, and in the command arguments type <“input filename”. The < sign is for redirected input and the input filename is the name of the input file (including the path if not in the working directory).

If you use macOS or linux (or windows using powershell), you may use the terminal to compile and run your program using g++. This is the compiler that we use in the autograder on GradeScope (you may need to install g++ on your computer). Once you installed g++, you may compile your program using the following command:

```
g++ project1.cpp -o p1
```

You may then run the program with the redirected input using the following command:

```
./p1 < input1.txt
```

Make sure your program and your input file are in the same folder.

A simple program that reads the input file and displays everything that was read is given below.

```
#include <iostream>

using namespace std;

int main ()
{
     int numNodes, maxEdges;

     cin >> numNodes >> maxEdges;
     cout << numNodes << maxEdges;

     return 0;
}
```

Submission:

Submission will be through GradeScope. Your program will be autograded with test cases and then hand graded to check for documentation and if you have followed the specifications given. You may submit how many ever times to check if your program passes the test cases provided. Test cases will be released at the beginning and later other test cases will be released while grading. For the autograder to work, the program you upload must be named as **project1.cpp**.

Your final submission must contain your source file named **project1.cpp**. You access GradeScope using the tab on the left in our course page on canvas.

Sample output file for corresponding input files will be released. The input1.txt file given is a simple input file that will help you understand the project, more complicated ones will be released later and used for grading.

<u>Constraints:</u>

3. In this project, the only header you will use the header files that are given to you in the boiler plate code. Using other or excess header files will be subject to a heavy grade penalty for the project.
4. None of the projects is a group project. Consulting with other members of this class on programming projects is strictly not allowed and plagiarism charges will be imposed on students who do not follow this.
    - Please review academic integrity policies in the modules.

<u>How to ask for help:</u>

4. Check FAQ discussion board of the project first.
5. Email the TA with your precise questions.
6. Upload your well commented program to CodePost.
    a. This is a website which is used to share code.
    b. You will upload your program and I can view it and comment on it.
    c. Here is the <u>invite link</u> to our class for the summer.
    d. Once you join the class on CodePost, you can upload your program to the Project 1 assignment.

        Note: Your program will <u>not</u> be auto graded at CodePost, this is just for when you want to ask a question and a place where I can look at your program and comment on it. CodePost is great for live feedback. GradeScope is the place where you should submit and where your program will be autograded.