**Department of Computing and Information Technology**
**COMP 2611 – Data Structures**
**2023/2024 Semester 1**
**Assignment 2**

*Date Due:* *Sunday November 26th, 2023 @ 11.55 pm*

## Overview

This assignment requires you to write a program that reads the information for a graph from a file and creates the graph in memory using an adjacency list. Your program must then read commands from another file, `Commands.txt`, and perform the corresponding operations on the graph.

## Requirements

1. Unzip the `Assignment2-Files.zip` file. You will get a Dev-C++ project, `Assignment2.dev` containing one .h and two .cpp files.

| **Filename** | **Description.** |
|---|---|
| `Graph.h` | Declaration of structs for a Graph and the functions that operate on a Graph. |
| `Graph.cpp` | Partial implementation of Graph functions. |
| `Assignment2.cpp` | Reads the information for a graph from a file and creates the graph in memory. It then processes the commands in the `Commands.txt` file. |

2. In `Graph.cpp`, write the code for the graph functions listed in Table 1.

| Return Type | Prototype of Function and Description |
|---|---|
| `Vertex` | `newVertex (string ID)`<br><br>Creates and returns a new *Vertex* struct with the given ID. |
| `Edge *` | `newEdge (string destID)`<br><br>Creates and returns the address of a new *Edge* struct with the given destination ID. |
| `int` | `findVertex (Graph * graph, string ID)`<br><br>Returns the location of the vertex with the given ID in the graph or -1 if it is not present. |
| `bool` | `addEdge (Graph * graph, string sourceID, string destID)`<br><br>The *addEdge()* function checks that the *sourceID* and *destID* are valid vertices in the graph. If so, it creates an edge to *destID* and adds the edge to the vertex, *sourceID*. The edge should be inserted in sorted order. The function should return *true* if the edge was successfully created and added to the graph and *false*, otherwise. |
| `bool` | `hasEdge (Graph * graph, string X_ID, string Y_ID)`<br><br>Returns *true* if there is an edge between *X_ID* and *Y_ID* in the graph, and *false*, otherwise. |
| `bool` | `deleteEdge (Graph * graph, string X_ID, string Y_ID)`<br><br>Deletes the edge between *X_ID* and *Y_ID* in the graph. Returns *true* if the edge was successfully deleted, and *false*, otherwise. |
| `bool` | `hasVertex (Graph * graph, string X_ID)`<br><br>Returns *true* if the graph has a vertex, *X_ID,* and *false*, otherwise. |
| `void` | `displayGraph (Graph * graph)`<br><br>This function displays the graph on the monitor, using the same layout that is used to store the information about the graph in a file. |
| `int` | `outDegree (Graph * graph, string X_ID)`<br><br>This function returns the out degree of vertex *X_ID*, if it exists. If the vertex does not exist, it should return -1. |
| `int` | `outgoingEdges (Graph * graph, string X_ID, string outgoing[])`<br><br>This function copies all the vertices adjacent to vertex *X_ID* in the graph (i.e., outgoing edges) into the *outgoing* array passed as a parameter. It returns the number of vertices adjacent to *X_ID* (i.e., its out degree) or -1, if *X_ID* does not exist in the graph. |
| `int` | `incomingEdges (Graph * graph, string Y_ID, string incoming[])`<br><br>This function copies all the vertices *X_ID* such that an edge (*X_ID*, *Y_ID*) exists (i.e., the incoming edges of *Y_ID*) into the *incoming* array passed as a parameter. It returns the number of vertices in the array (i.e., the in degree of *Y_ID*) or -1, if *Y_ID* does not exist. |

**Table 1: Graph Functions (to be written in Graph.cpp)**

3. In `Graph.cpp`, write the code for the *readGraph()* function with the following prototype:

```
Graph * readGraph (char fileName[])
```

The function should open a text file specified by the character array, *fileName*, read the information from the file, and create the graph. An example text file is given below:

```
7
A B C D E F G
A 2 D G
B 0
C 2 E G
D 4 B E F G
E 1 G
F 3 B E C
G 0
```

The first line of the file indicates that there are 7 vertices. The second line is the name of each vertex. Each subsequent line is for one of the 7 vertices (in order). It lists the vertex and then indicates how many edges leave that vertex. For example, there are two edges out of *A* and none out of *B*. The remaining data on the line are the vertices to which the given vertex is connected. For example, *A* is connected to *D* and *G*.

4. In `Assignment2.cpp`, write code to process the commands from the Commands file, `Commands.txt`. Each line in the command file contains one of nine (9) commands. The commands are described in Table 2.

| Command | Command Followed By | Description |
|---|---|---|
| 10 | file name (string) | Create the graph from the information specified in the given file. |
| 11 | vertex name (string) | Find out if the vertex is present in the graph. |
| 12 | vertex name (string) | Find out the number of edges leaving the vertex in the graph. |
| 13 | vertex name (string) | List all the vertices that are adjacent to the given vertex. |
| 14 | vertex name 1 (string)<br>vertex name 2 (string) | Find out if there is an edge from vertex 1 to vertex 2 in the graph. |
| 15 | vertex name (string) | List all the vertices *X* for which there is an edge **from** *X* **to** the specified vertex. |
| 20 | vertex name 1 (string)<br>vertex name 2 (string) | Insert an edge between vertex 1 and vertex 2. |
| 21 | vertex name 1 (string)<br>vertex name 2 (string) | Delete the edge between vertex 1 and vertex 2 in the graph. |
| 30 | Nothing | Display the graph on the monitor. |
| 99 | Nothing | Terminate the program. |

**Table 2: List of Commands in the Commands File**

## Data Files

A graph file, `Graph.txt`, and a Commands file, `Commands.txt`, have been provided with the assignment. Final versions of these data files will be made available closer to the deadline. Sample output that is expected from your program using the final versions of the data files supplied will also be made available.

## What to Submit

Create a zipped file containing all the files in the project as well as the data files. Upload the zipped file to myeLearning on or before the deadline.

## Programming Guidelines

You may add .h and .cpp files to the project. You may also add functions to the given .h and .cpp files.