COMP 2611 – Data Structures
2018/2019 Semester 3
Assignment #2

Date Due: 30th June, 2019 @ 11:55p.m.

Description

You are given a Dev C++ project, *Assignment3*, which contains code that does the following:

- Reads a graph stored in a text file using the format specified in the text book. Each edge in the graph has the distance from an origin airport to a destination airport. A sample file, `graph.txt`, shows the format that should be used (from Page 245 of the text book).
- Stores the graph in an adjacency list.
- Finds the shortest path from a given vertex to the other vertices of the graph using Dijkstra's algorithm.

The code for the min-priority queue which is required by Dijkstra's algorithm is also included.

Write the code for the functions listed below in `Graph.cpp`. Note that the `Graph.h` file already has the prototype for the functions.

Function and Description
`bool deleteEdge (Graph * graph, string u, string v):` Removes the edge between *u* and *v* from the graph, if it exists. Returns *true* if successful, and *false* otherwise.
`bool hasEdge (Graph * graph, string u, string v):` Returns *true* if there is an edge between *u* and *v* in the graph, and *false* otherwise.
`bool hasVertex (Graph * graph, string v):` Returns *true* if there if there is a vertex *v* in the graph, and *false* otherwise.
`int outDegree (Graph * graph, string v):` Returns the amount of edges leaving *v* in the graph, i.e., the out-degree of *v*. If *v* does not exist, returns -1.
`int adjacentTo (Graph * graph, string v, string adj[]):` Copies all the vertices adjacent to vertex *v* in the graph into the *adj* array passed as a parameter. It returns the amount of vertices adjacent to *v* (incidentally, this value is the same as the out-degree of *v*).
`void printAllPaths (Graph * graph, string sourceID, string destID):` Given an origin airport and a destination airport, finds all the possible paths from the origin airport to the destination airport.
`void depthFirstTraversal (Graph * graph, int source):` Performs a depth-first traversal of the graph. All the vertices must be visited regardless of the source vertex.
`void breadthFirstTraversal (Graph * graph, int source):` Performs a breadth-first traversal of the graph starting from the source vertex.

Main Program

Your main program should provide a menu from which various operations are performed. The operations are performed by calling one or more of the functions from the table above.

The following is the menu that must be displayed:

```
Assignment 3: Working with Graphs
-------------------------------------------------

1. Create Graph from File
2. Does the Airline Fly from Location A?
3. How Many Direct Flights from Location A?
4. What are the Direct Flights from Location A?
5. Is There a Direct Flight from A to B?
6. Delete Flight from A to B
7. Print All Paths from A to B
8. Shortest Path from A to Other Destinations
9. Perform Breadth First and Depth First Traversal
Q. Quit

Please enter an option:
```

When an option is selected, the appropriate action must be taken, after which the menu is re-displayed. The following is a description of each option.

1. When this option is selected, your program should allow the user to specify the name of the file containing the graph:

```
Please enter the name of the file or M (Menu):
```

If the user enters "M" control should return to the main menu and no graph should be created; if a graph was previously created, this will continue to be the "active" or "current" graph. Otherwise, your program should read the given file and create the graph.

2. When this option is selected, your program should prompt the user for an origin and determine if there are any flights leaving this origin.

3. When this option is selected, your program should prompt the user for an origin and determine how many flights leave this origin.

4. When this option is selected, your program should prompt the user for an origin and list all the flights that leave this origin.

5. When this option is selected, your program should request the user to specify two locations:

```
Please enter the origin and destination M (Menu):
```

If the user chooses "M", control should return to the main menu. Otherwise, your program should determine if there is a direct flight between the origin and the destination.

6. When this option is selected, your program should request the user to specify two locations:

```
Please enter the origin and destination M (Menu):
```

If the user chooses "M", control should return to the main menu. Otherwise, your program should delete the direct flight between the origin and the destination, if it exists.

7. When this option is selected, your program should request the user to specify two locations:

```
Please enter the origin and final destination or M (Menu):
```

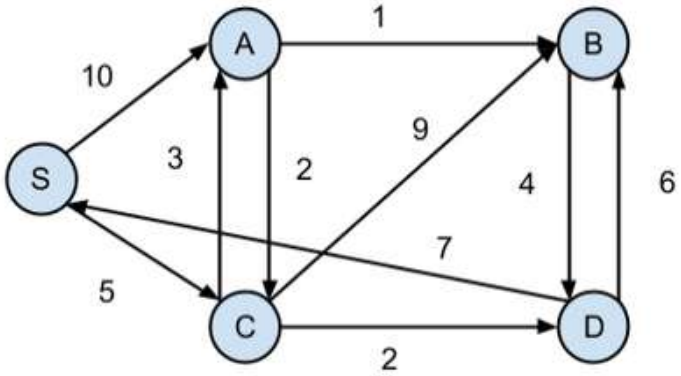
If the user chooses "M", control should return to the main menu. Otherwise, your program should determine if it is possible to travel from the origin to the destination and print all paths from the origin to the destination.

8. When this option is selected, the program should prompt the user for an origin and display the shortest path from the origin to all the other destinations, provided these paths exist. Dijkstra's algorithm should be used for this option. You are allowed to make modifications to the code in `Dijkstra.cpp` to achieve the desired output.

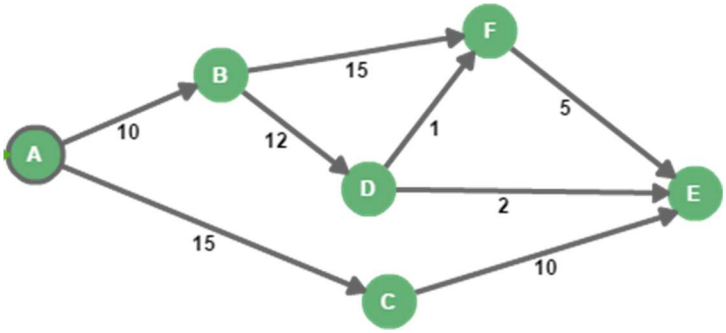
9. When this option is selected, your program should prompt the user for an origin and perform a breadth-first traversal starting from the origin as well as a depth-first traversal of the graph (all the vertices must be visited in the depth-first traversal).

Some sample graphs are given in the next page. You should create a text file with the data for **each** of the four graphs.

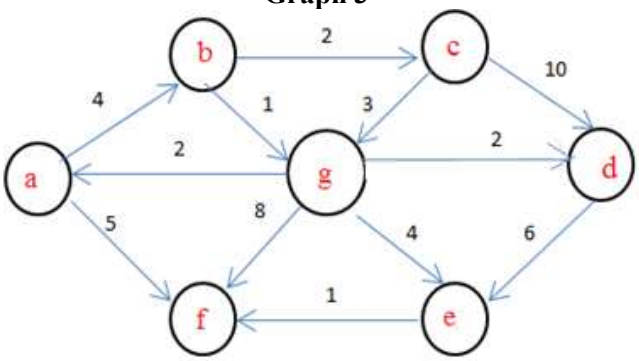
Graph 1



Graph 2



Graph 3



Graph 4