Assessed Exercise week 6

**Before attempting the tasks, please check the “hints” you can find below each task.**

**Task A: Graphs**

Implement a **Windows Forms Application (.NET Framework)** with a GUI which allows the user:

* to add a node to a graph (given the ID of the node)
* to add a directed edge between two nodes (given the IDs of the two nodes)
* to display the total number of nodes in the graph
* to display the total number of edges present in the graph

Assume that the IDs of the nodes are int.

*Hints: We have seen the implementation of classes Graph and GraphNodes in the lecture /lab (you can find the code of these two classes from the lab slides on Moodle or from the lab document on Moodle). The methods to count the number of nodes and edges are lab exercises. In week 2 we have seen how to add a data structure to a GUI (“stack in a GUI”); here you can use the same idea – instead of adding a stack you should add a graph to the GUI. The input (which are IDs of nodes) can be read from textboxes (as strings). Output can also be displayed on textboxes or labels. Remember that any input read from a textbox is a string and one can convert a string s to an int using the instruction Convert.ToInt32(s). On the other hand, to display an int i in a textbox, one needs to convert the number i into a string and this can be done as: i.ToString().*

*See also the “getting started“ video on Moodle, that you can find just below the spec of the exercise.*

**Task B: Airports Management**

Implement a **Windows Forms Application (.NET framework**) in C# (with a GUI) which allows users to:

* Insert an airport by using its code (assume the airport code is a *string*, see Figure 1 below)
* Insert / Remove a direct connection (i.e., direct flight) between two airports given their codes (see Figure 1 below)
* Display the codes of the airports that can be reached from a *starting airport* by either using direct or connecting flights. In this case, allow the user to insert the starting airport and specify if interested in *only direct* or *both (direct and connecting)* types of flights.

In your screencast, test the application on the airports shown in Figure 1 (below).

*Hints to complete the task: To implement the application you will need to use and adapt the classes (and the methods) of Graph and GraphNode seen in the lecture/lab to be able to store the code of the airports (string) as ID of the nodes. To implement the removal of an edge, check how we did the addition of an edge. For the last point (“Display the codes of the airports that can be reached…”) use the idea of graph traversal. Start from the starting code for the DFS that you can find on the lab slides (remember that in this task the ID of a node is a string). The method to implement the DFS needs to be in the class Graph. On the GUI you may use a checkbox to allow the user to specify if interested in only direct* or *both (direct and connecting) types of flights.*

*Figure 1*

*The graph shows a set of airports (identified by their* ***code****, which is a string) and the established* ***direct*** *connections (i.e., representing direct flights). A direct connection (i.e., direct flight) between the two airports is denoted by a* ***directed*** *edge. e.g., there is a direct flight going from cdg to lis, but there is no direct flight going from lis to cdg.*

*Airport codes (cgd, lis, etc..) are strings.*

*Starting from lux (i.e., starting airport) one can reach (by using direct or connecting flights): ams, cdg, lis.*

*On the other hand, starting from lux (i.e., starting airport) one can reach (by using only direct flights): ams.*