# 6G5Z0024

**Assessed Exercise 4**

**This assessed exercise has 3 assessed tasks: Task A, Task B and Task C**

**Task A**

Graphs can be utilized to model various kinds of scenarios. For this assessed task, we will make use of the graph structure that has been discussed in the lectures and lab material to model a social network of individuals. We will assume that a directed edge represents an individual sending a message of sorts with another individual present on the network.

**An example of this network is available below as Figure 1.**

Implement a graph-based structure (which is based on the provided material) that consists of the **GraphNode** and **Graph** classes.

Use these classes when implementing an application with a user interface. This application should allow a user to:

* **Insert** a new individual as a **node** into a graph (the name of the individual can be used as the ID for your new node).
* **Insert** a **directed edge** between two nodes (given their IDs).
* **Display** the **current number** of individuals (**nodes**) that are in the graph.
* **Display** the **current number** of **edges** that are in the graph.

The user interface here can be a GUI (Windows Forms Application) or a menu driven interface (Console Application).

*Hints to complete the task: When you have created your new project, you should begin by implementing the* ***GraphNode*** *and* ***Graph*** *classes that have been discussed in the lab material.*

*On Moodle, you can find the lecture slides that discuss the basic ideas (graphs, adjacency list) necessary to solve this task.*

*Notice that in this implementation, the graph is represented using an adjacency list implemented using the built-in LinkedList from C#. We are going to discuss and implement many of these methods in the lecture/lab – so please check the slides of the lecture and lab when completing this task as many of the methods are implemented there.*

**Task B**

When implementing graphs, we can define additional rules to more closely model a given scenario. We can also implement additional functionality to obtain additional details from these networks.

Expand on Task A by adding functionality for:

* Allow for **edge weights** to be added **when creating a new edge** within the networks, here edge weights will represent the frequency of an individual sending messages to another individual (this can be an **integer** value between 1 - 10).
* Create a function that can display the following to the user:
  + The **average number** of outbound **connections** between individuals.
  + The **average weight of edges** within the network.
  + Which **individual** has the **most outbound connections**?
  + Based on edge weights, which individual communicates, **on average**, the most frequently with their friends?
* Given the name of an individual, display all the names that individual has sent messages to.

*Hints to complete the task: To implement edge weights into your graph, consult the lab materials from Week 10 to aid you with integrating this into your structure.*

*For the network information, we have discussed how we can write a function to obtain connection averages in the lab material, expand on this for implementing each of the additional methods here. You may wish to write a function into your graph class for each network metric to be obtained which can be used give feedback about the network to the user either as requested or when the network is updated.*

*For displaying connected friends, see Figure 1. If a user requested the see all of the friends Tim is connected with, the output here should be along the lines of ‘Cassandra, Mavis’*

**Task C**

Further expand on A and B by including functionality for:

* Removing an individual from the network.
* To understand how a rumor can propagate across the network, implement a function that displays the names of all individuals that can be reached on the network with the given name of an individual as a starting point.

*Hints to complete the task:*

*When removing an individual from a network, what else needs to be considered besides the node? In figure 1, if Sam is removed, what else needs to be addressed?*

*For the second point, make use of graph traversals. Two have been discussed in the lecture/lab material, DFS & BFS. You may use either of these traversal functions to address the second point of this task.*

**Figure 1**

