Coursework Specification

Please read the entire coursework specification before starting work.

This is an individual project development coursework.
You are expected to work on your own to implement the project.

Overview

The overall task for this project is to design and develop a standalone application in Python for analysing small social networks and recommending likely new friends.

The project is an extensive application that needs to be developed and tested incrementally. The implementation requires the design, implementation, and testing of an object-oriented Python program that follows the specification below and uses custom classes (classes you have defined), appropriate data structures, and file processing. The specification below provides less overview on how to do the project as you are expected to come up with and design the solution yourself.

Your project development is the culmination of all you have learnt about Python and OOP and will showcase all your hard work for part one of this module. The system you develop should become a strong addition to your programming portfolio. Hence, you should aim to produce a system that is well-designed, robust, and useful. The system should operate smoothly without sluggishness or crashes and should not require instructions or a manual to use.

Scenario and System Specifications

Social networks are made up of a set of members and connections (dyadic ties) between those members - for example, there exists a connection between you and everyone whom you befriend/interact with within a social network. Facebook and WhatsApp are among the many popular social networks with which you may already be a member. An interesting feature of these social media sites is they automatically recommend likely new friends (connections) for you and other members of the network. They use sophisticated algorithms to do so. However, to better understand how this feature works, your project is to simulate a social network, analyse connections between members, and implement a naive method to recommend the most likely new friend(s) to members based on the intersection of their common friends. That is, the person recommended for member X as a friend is the person who has the most friends in common with member X, but who currently is not friends with member X. Your project will also analyse connections between friends and produce statistics as defined in the requirements section.

The program will need to map the connections (friendships/relations) between the network members by using appropriate data structure(s) to represent how members relate to each other. For simplicity, the only information you need about each member is their name (or member ID). The map will represent the member along with the connections to other members (the friends each member has). The input to your program is a file containing this map representation and the output described in the requirements section should be displayed on the PyCharm console. Your program should work via the console and does not need to create a graphical user interface – Graphical User Interface (GUI). There are no marks for creating a Windows-based application.

System Features and Requirements

The system must provide the following features and comply with the System Specifications outlined above. All the features below must be completed individually. All network member **input and output should be via the PyCharm console** and not via a GUI. The input must be validated for correctness, must prompt an error message if incorrect values are entered, and continue to ask for valid values until one is entered or "n" is entered to stop.

Task 0 – Planning and Setup (to be shown to your lab tutor as soon as possible)

i. **Create a plan** of how you will complete this project. The plan should show each of the features and sub-features listed below and when you expect to complete them.

ii. **Create a text file** called nw_data1.txt containing a representation of a social network in the following format (which will be used as input to your program to test your code and should be read on start-up):

- The first line of the file is an integer representing the number of members in the social network, then
- the following lines are in the form of *member* and *friend* pairs, where *member* and *friend* are represented by a member name OR member ID. The member/friend pairs should be listed on separate lines in the file.
- There is no limit to the number of members or relations between them in the social network. The only restriction is that number of member must match that determined by the integer on the first line of the file. For simplicity, start with smaller networks and gradually test your program with larger ones.

Sample social network and the lines that should show in the `nw_data1.txt` file:

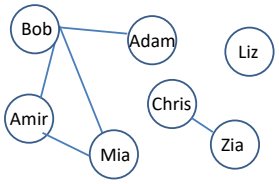
```
7
Adam Bob
Bob  Amir
Bob  Mia
Chris    Zia
Mia  Amir
Liz
```

or

```
7
0    2
2    1
2    4
3    5
4    1
6
```

The above is a representation of a social network that has 6 members with the following relations:

`Adam` is friends with `Bob`
`Amir` is friends with `Bob`, `Mia`
`Bob` is friends with `Adam`, `Amir`, `Mia`
`Chris` is friends with `Zia`
`Mia` is friends with `Amir`, `Bob`
`Zia` is friends with `Chris`
`Liz` is friends with `none`



Note: the network graph is for illustration only and you are not expected to draw it in your code.

iii. **Consider/design the classes, methods, and data structures needed** before starting to code. This coursework requires you to implement your program using classes so, start by considering suitable classes first. The think about the methods you will include in each based on the feature requirements below. Note that any program using classes can be written without classes, but there will be no credit for such solutions!

iv. **Create additional test input files** `nw_data2`, `nw_data3`, etc. containing larger social networks which you can later use to test and validate your program.

F1 – Feature 1: Social Network Data

i. **Open social network file**:
this sub-feature should allow the user to enter a file name and prompt an error message if the file cannot be opened. The process should continue to as ask for a valid file name until the file can be opened or the user types "n".

ii. Simulate the social network structure:

- **Get social network data**:
  once the file is opened successfully, read the file (following the format described in `nw_data1.txt`) and dynamically create an appropriate data structure named `social_NW` to hold the social network so that it can be used elsewhere in the code;
- **Display social network:**
  ask the user whether they want to display the social network read from the file. If they do, pretty print (display nicely formatted) the social network on the **PyCharm console**.
  An example of the output is as follows:

```
Adam  -> Bob                          0 -> 2
Amir  -> Bob, Mia                     1 -> 2, 4
Bob   -> Adam, Amir, Mia      or      2 -> 0, 1, 4
Chris -> Zia                          3 -> 5
Mia   -> Amir, Bob                    4 -> 1, 2
Zia   -> Chris                        5 -> 3
Liz   ->                              6 ->
```

F2 – Feature 2: Friend Recommendation

The aim is to recommend the most likely new friend(s) to an existing network member based on the intersection of their common friends. The algorithm for this recommendation is as follows: for each member of the social network, count the number of friends they have in common with each of the other members. Then for each member in the network, the friend that is recommended for them is the member of that network whom they are not currently friends with but have the most friends in common. *It makes sense, intuitively, why they may like to be connected as friends*. In this simulation, members with no friends will NOT be recommended any new friends.

i. Get a common friend count for each member of the social network:

- This should return the number of friends that any pair of members in the social network have in common.
- the count must be produced for every member in the network and maintained accordingly in an appropriate data structure (the data structure must represent all possible member pairs in the social network along with the count of common friends between them). Return the data structure as output and name it, for example, `common_friends`.

  Hint: to approach this, you must:
  1. first work out the number of common friends between 2 members of the network
     e.g. given the sample network in `nw_data1.txt` the number of common friends between `Adam` and `Amir` is `1`, between `Bob` and `Adam` is `0`, and between `Bob` and `Zia` is `0`.
  2. do the same for all the pairs of members in the network and include the resulting numbers in the `common_friends` data structure.
- Example pretty print of `common_freinds` given the sample network in `nw_data1.txt` is:

```
Adam ->[1,1,0,0,1,0,0]                0 -> [1,1,0,0,1,0,0]
Amir ->[1,2,1,0,1,0,0]                1 -> [1,2,1,0,1,0,0]
Bob  ->[0,1,3,0,1,0,0]      or        2 -> [0,1,3,0,1,0,0]
Chris->[0,0,0,1,0,0,0]                3 -> [0,0,0,1,0,0,0]
Mia  ->[1,1,1,0,2,0,0]                4 -> [1,1,1,0,2,0,0]
Zia  ->[0,0,0,0,0,1,0]                5 -> [0,0,0,0,0,1,0]
Liz  ->[0,0,0,0,0,0,1]                6 -> [0,0,0,0,0,0,1]
```

  Note: in this example, `Liz` has no friends in the social network as we discount members being friends with themselves.

ii. Recommend a friend:

- Given a member name (`m_name`) or ID (`m_ID`), determine the largest common friends count for that `m_name`/`m_ID` from the `common_friends` structure produced in F2.i and return the corresponding member for that value as the most likely friend. That corresponding member is the most likely friend because by having the largest value in the row, it means it is that member who has the most friends in common with the `m_name`/`m_ID`. Of course, you don't want to recommend someone who is already a friend, and it also doesn't make sense to recommend the `m_name`/`m_ID` as their own friend.
- Example output given the sample output from 1:

```
Recommended friend for Mia is Adam   or  Recommended friend for 4 is 0
Recommended friend for Bob is none   or  Recommended friend for 2 is none
```

F3 – Feature 3: Social network analysis and statistics

i. Display how many friends a given member has:

- This sub-feature should allow the user to enter a member name or ID and display the number of friends that the member has. It should validate that the input exists and prompt an error message otherwise.

ii. Show the members with the least number of friends and those who have no friends at all:

- Display all the members with the least number of friends and those with no friends at all. Pretty-print your output on the PyCharm console.

iii. Show the relationships for a given member:

- This sub-feature should allow the user to enter a member name or ID and show all the friends that the member is connected within the social network. It should validate that the input exists and prompt an error message otherwise. The output should display the relationships for that member.

iv. Find indirect relationships (connections) between members:

- For every member in the social network find the friends of all the **direct** friends of that member but do not include the member themself. Friend-of-a-friend relationships are indirect connections between people and represent indirect relations between members. This sub-feature should return an appropriate data structure with those connections and name it as indirect_friends The data structure only needs to represent the first level of indirect relations (i.e., only contain the friends of direct friends of each member). Pretty-print the elements of the data structure on the PyCharm console.

v. Validate the input:

- Update F1.ii to validate the social network data read from the file after it has been opened successfully. This sub-feature should check the consistency of friendships (connections) in the network. The network representation is considered consistent if member A is friends with member B AND member B is also friends with member A. The network representation is inconsistent otherwise. If the network representation is found to be inconsistent, raise an exception and prompt the user with an error message.