Program 5 - Social Networking!

Due Date: **May 3rd @ 6am**; Late Programs excepted no later than **May 5th @ 6am** for a 30 point penalty.

Assignment Overview

This assignment will give you more experience on the use of LinkLists.

Have you ever wondered how social media sites suggest friends to you? Well, most of the sites do indeed use highly sophisticated algorithms, but for this project you will implement a naive method to recommend the most likely new friend to users of a social network.

Background

A social network such as Facebook consists of a set of users and connections between the users (i.e., there exists a connection between you and everyone who you have befriended). An interesting problem is to write a computer program that can automatically suggest possible new connections (i.e., friends) for each user.

For this project you will implement a system that will, for each user, suggest the most probable user to befriend based upon the intersection of your common friends. In other words, the user that you will suggest to Person A is the person who has the most friends in common with Person A, but who currently is not friends with Person A.

The algorithm begins with: *for* each user, go through all the other users and calculate the number of friends they have in common. Then for a given user the friend you will suggest to them is the user in the social network who they are currently not friends with, but have the most friends in common. Intuitively, it makes sense why they might want to be connected as friends. (Note that when you look for your most common friend in this scheme it will be you, i.e. you will have to remember to remove yourself from consideration.)

Project Description / Specification

Two files have been provided for you to run your program with. One is a small test file containing a made up set of users and connections between them (`small_network_data.txt`). The other is a subset of a real Facebook dataset (`facebook_1000_data.txt`), which was obtained from https://snap.stanford.edu/data/egonets-Facebook.html.

The format for both files is the same:

- The first line of the file is an integer representing the number of users in the given network (`n`).
- The next `n` lines of the file contain the names of the users. Each user is made up of a `name` and an `id` which is an integer between `0 - (n-1)` .
- The rest of the lines are of the form: `user_u user_v` , where `user_u` and `user_v` are the IDs of the two users who are friends.

For example, here is a very small file that has 5 users in the social network (`small_network_data.txt):

```
5
Amy
Bill
Erik
Judy
```

```
Liz
0 1
1 2
1 4
2 3
```

The above is a representation of a social network that contains 5 users. User ID=0 is friends with User IDs = 1 User ID=1 is friends with User IDs = 0, 2, 4 User ID=2 is friends with User IDs = 1, 3 User ID=3 is friends with User IDs = 2 User ID=4 is friends with User IDs = 1

Although this could be implemented in many different ways, you are required to design a social network consisting of a `LinkedList` of `Friend` objects. You will have a `Friend` object for each user - made up of a name and ID. Each `Friend` object will also have a vector of integers representing his/her friends.

Implementation Details

You must implement the following functions within your `main.cpp` :

1. `open_file()` - Attempts to open the file listed on the command prompt. The program will try to open the data file. An appropriate error message should be shown if the data file cannot be opened. and the program will end; otherwise, the function returns the fileobject.
2. `read_file(ifstream&)` - Will have the file object passed to it (by reference). The function will then dynamically create a linkedlist which must be ON THE HEAP using dynamic memory allocation. You will read in each user and create a `Friend` object with each user and add it to the LinkedList. Then you will read the friends network information and add to the appropriate users friend list (vector of ids). **After the file has been completely read, and the network created (i.e. the linked list) you will return a pointer to the linked list that was dynamically created.**
3. `recommend_user(LinkedList *network)` - This function accepts a pointer to your linked list (network) and will be used to prompt the user for an ID of a user to recommend a friend for. g) Note that when prompting the user for the user_id you should validate that the input is in the correct range, which is `0 to n`. Once you receive the id, you will call the `get_recommendation()` function of your LinkedList network to return the appropriate friend object, then display the ID and name of the recommended friend.
4. `main()` - In the main method, you will need to first call `open_file()` , followed by calling `read_file(fp)` . After obtaining the network, you will need to have a while loop that will repeatedly ask the user for a user_id that should be in the range of [0,n-1] (i.e., from 0 to n-1, inclusive) using your `recommend_user()` function from #3.
5. After displaying the suggested friend for the user that had their id as user_id you should prompt the user if they want to see the suggestion for another user input or if they want to exit. To exit the program you should accept all forms of the string ‘no’ (i.e., ‘NO’, ‘No’, ‘nO’, and ‘no’).

The Linked List (Network)

Your Linked List should be implemented as a Singly Linked List and have the following functions. *NOTE: You may not need all of these functions for your final submission, however we still expect you to implement and test that they work correctly.*

1. `add_user(friend)` - a function that will be used to add a user to your linked list
2. `add_friend(int u, int v)` - a function that will add the friend ID `v` to the friends vector of the friend object with ID `u` .
3. `display_network()` - a function that is used to display the network in a readable format (see output below for inspiration). This will be helpful when debugging your program!

4. `get_recommendation(user_id)` - a function that accepts a users id and determine a recommendation for that friend. Once a recommendation is found, the function returns a `Friend` object of the recommended friend. Your function should determin the recommendation based on who has the most # of friends in common with the user in question. It might help to write out this algorithm by hand first using *psuedocode*.

Example Output

Example output of a `display_network()` function: (using `small_network_data.txt`)

```
0 (Amy): [ 1 ]
1 (Bill): [ 0 2 4 ]
2 (Erik): [ 1 3 ]
3 (Judy): [ 2 ]
4 (Liz): [ 1 ]
```

Example output of program:

```
Welcome to the Social Network!

Reading in the network file...

Successfully read in network file!

Enter an integer in the range 0 to 4: abc
Error: input must be an integer between 0 and 4
Enter an integer in the range 0 to 4: 0
The suggested friend for 0 is 4

Do you want to continue (yes/no)? Yes
Enter an integer in the range 0 to 4: 3
The suggested friend for 3 is 1

Do you want to continue (yes/no)? No

Goodbye!
```

NOTE: Notice that 4 and 2 are both valid friend suggestions for userId 0 for the above scenario

Submission Details

You must submit your program via GitHub by **May 3rd @ 6:00am**.