Assignment 3: Graffit 1.1

Now that we're all socially distanced (and for a few more months from the looks of it), it is even more important that we connect with one another through online platforms. Social media platforms like Facebook, Twitter, Snapchat, etc. all offer ways for you to connect with others. In the simple case, they allow you to connect with other human users, with brands, and with communities (think Facebook group chats or r/UTSC). These topics are connected into an intricate web of social connections, a social network! So, here's an assignment all about these social networks, at the CSCA48 level. (You wish programming for these social media companies was this easy!)

Let's work with the now official CSCA48 social media network: Graffit. In Graffit, we have users (like *you* and *me*) and brands (like *UTSC*). **Users** will want to keep track of their friends and the brands they like. **Brands** would like a list of brands similar to themselves. We will be manipulating a network with these kinds of structures, so let's Graffit!

You will be responsible for implementing a few functions…

For **Users**, we should be able to…

- create_user(char[] name)
    - Each name will be unique.
- delete_user(User user)
- add_friend(User user, User friend)
    - Friends here are a two-way connection. If you're my friend, I have to be your friend. Order these alphabetically using strcmp.
- remove_friend(User user, User friend)
- follow_brand(User user, BrandNode brand)
    - Order these alphabetically.
- unfollow_brand(User user, Brand brand)
- get_mutual_friends(User a, User b)
    - Return an integer representing the number of mutual friends between user A and user B. A mutual friend is someone who is friends with both A and B.
- get_degrees_of_connection(User a, User b)
    - Distance between two users.

For **Brands**, we should be able to…

- connect_similar_brands(char[] brandNameA, char[] brandNameB)
- remove_similar_brands(char[] brandNameA, char[] brandNameB)

To help out a bit, the following functions are provided:

- in_friend_list(FriendNode *head, User *node)
- in_brand_list(BrandNode *head, char *name)
- insert_into_friend_list(FriendNode *head, User *node)

- insert_into_brand_list(BrandNode *head, char *node)
- delete_from_friend_list(FriendNode *head, User *node)
- delete_from_brand_list(BrandNode *head, char *node)
- print_user_data(User* user)
- print_brand_data(Brand* brand)
- populate_brand_matrix(char* file_name)
    - Turns an adjacency matrix represented in a txt file into a network of brands. This txt file will be formatted in a specific way:
        - The first row will be brand names. Each brand name will be unique and will be separated by one space character.
        - The following rows will create a NxN matrix. This will be a valid adjacency matrix, so the element at (x, y) = the element at (y, x). The values will be either 0 or 1 and will be separated by one space character.
    - Note how we will not be adding new brands as we go along, nor removing brands.
    - As a benchmark of your progress as a computer scientist, you should be able to understand what this code does.

The hard part...

Graffit wants to compete with all the big names in the game. Many social media companies have a "suggested friends" feature. We will build a function to just get a singular suggested friend. Return the not-already-friended user with the most brands they follow in common. If there's a tie, return reverse-alphabetically (z to a).

get_suggested_friend(User user);

Graffit is a company of the future, and "suggested friends" is a feature of the past. To stay ahead of competitors, Graffit pioneered the "add N suggested friends" feature where it automatically adds new friends for you! It also implemented the "follow N suggested brands" feature which automatically follows brands you might like. These will be implemented through the following functions:

add_suggested_friends(User u, int n)

Same rules as the get_suggested_friend(...) function. If there are less than N possible options to add, add as many as possible. Hint: Complete *get_suggested_friend* first and this should fall into place.

follow_suggested_brands(User u, int n)

A suggested brand is the brand with the most similarities with brands the user already follows. If there's a tie, the suggested brand is the one that comes first reverse-alphabetically (z to a).

As always, proper and reasonable error checking is expected.