Programming Assignment 3

Assigned: Oct. 19
Due: Nov. 7

In this assignment, you will redo the same MicroFB project from assignment 1, but this time using hash tables and linked lists to allow an open-ended number of friends and gain efficiency.

The input and output is identical to assignment 1, except that there is an additional command for unfriending.

- U ⟨name1⟩ ⟨name2⟩ — Record that the two specified people are no longer friends.

However, this time, unlike assignment 1:

- All these commands should execute in time that is independent of the total number of people in the system.
- The command "L" should take time proportional to the number of friends that the person has. The command "U" should execute in time proportional to the sum of the friends that the two named people have. The commands "P", "F", and "Q" should execute in constant time.

Data Structures

You *must*

A. Define a "Person" class which has one field for the name and another field for the linked list of friends.

B. Store the friends of each person in a linked list. You may use any linked list class you want: The Java library linked lists, a linked list class you write yourself, any of the linked list classes presented in class, something you found somewhere on the Web, etc. However:

   – The list of friends *must* be in a linked list of some description, and *not* in an array.
   – If you use any class that is not a Java library class and is not of your own design then you *must* indicate its source in a comment. This applies to the linked list classes presented in class as well.
   – The list must be a list of `Person` objects, *not* a list of their names, as `Strings`.

C. Create a global hash table `allPeople` that indexes each `Person` object under the name. Here, I recommend that you use the Java library functions; but you may use a hash table definition from anywhere else. The same rules apply on citations.

D. In order to execute the command "Q" in unit time, you will need a second hash table `allFriends`, that uses as a key a pair of names. The way you use a pair of names as a key is to put the two names in alphabetical order, then combine them separated by an asterisk. For instance, the pair of names "Sam" and "Liza" becomes the string "Liza*Sam". This second hash table uses these strings as keys, and booleans as values.

Executing commands

To execute a "P" command, create a `Person` object for the name, and save it in `allPeople` under the name.

To execute an "F" command:

- Find the two `Person` objects in `allPeople`.
- Construct the two person key for the two names.
- Check in `allFriends` that these people are not already friends. If so, exit.
- Add each person to the front of the linked list of the friends of the other person.
- Add the two person key to `allFriends`, with value `true`.

To execute a "U" command:

- Find the two `Person` objects in `allPeople`.
- Construct the two person key for the two names.
- Check in `allFriends` that the people are friends. If not, exit.
- Delete each person from the linked list of the friends of the other person.
- Delete the two person key from `allFriends`.

To execute an "L" command", find the person object in `allPeople` and loop through the list of friends.

To execute a "Q" command, construct the two-name key and look it up in `allFriends`.

Input/Output

The rules on error checking are the same as in assignment 1. That is, you may assume that the input is correctly formatted, but you should check for contentive errors:

- Creating a person with a name already in use.
- Referring to a non-existent person.
- Friending two people who are already friends.
- Friending oneself (i.e. the two arguments of F are the same).
- Unfriending people who are not friends.

In any of these cases, the program should do nothing, and not crash.

Submission

Upload the source code to NYUClasses. The "main" method should be in a class called `MicroFB2`, and therefore in a file called MicroFB2.java. If your code is all in one file, you may simply attach the .java file. If your source code is in several files, combine them into a single .jar file. Any instructions about how to run the code etc. should be in the body of the email message.