CS 104 (Fall 2013) — Assignment 11
Due: 12/12/2013, 11:59am

BitBucket directory name for this homework (case sensitive): HW11

This homework is the fourth and final part of your long-term Social Networking Site project. It will be graded as part of the project (rather than a regular homework), and thus cannot be dropped.

(1) Review Chapter 20 and the additional handout provided to you.

(2) In addition to the fields in your `User` class so far, add a field for a `username`. Your program should ensure that `username` is unique, i.e., no two users have the same `username`. (On the other hand, it's ok for multiple users to have the same name.) When a user logs in, they should now do so with their username and password.

(3) Ensure that all passwords are stored encrypted. To do so, implement either the MD5 or the SHA-1 hashing algorithm, and apply them to the user's password when he/she enters it, storing only the hash of the password. We will not go over these algorithms in lecture, so you are explicitly encouraged to seek out other sources (such as the Internet, or textbooks) to learn about them. When a user tries to log in, apply the same algorithm to the entered password, and let the user log in only when the hash of the entered password matches the stored hash. (This may allow users to log in with an incorrect password, but it's very unlikely that this happens.)

Also, if in any place in your application, you ever displayed a user's password, fix those places, and ensure that the password is in fact kept secret.

(4) Replace your `Userlist` with one of your own implemented `Dictionary` implementations from Homeworks 9 and 10. Build it so that `Userlist` allows searching by `username` and by real name. Since the `username` will be unique, you can just return a `User` or `User*`.

But the real name is not unique, and you may want to integrate this function with searching by partial strings, which you implemented before. Probably, the best way to implement this would be to return an iterator through all `User` (or `User*`) that match the string.

The lookup by `username` should be fast (i.e., implemented with a Hashtable or 2-3-Tree), while the lookup by real name (allowing only partial matches) can be slower — you probably cannot avoid linear search here without much more sophisticated data structures you haven't learned about.

(5) Implement the following functionality. A user who is logged in can enter the name (or username) of another user and find out how many degrees of separation they have (and how). In other words, if user u searches for user v, then the social networking site will find a shortest path between u and v and output all the nodes on the path (say, with their real names). If no such path exists, then that fact should be reported instead. If multiple shortest paths exist, then you only need to output one of them, but if you prefer, you can output all, or up to a fixed number, or up to a user-selected number.

(6) Come up with some cool new functionality for your social network and implement it. What you do is completely up to you. We will grade this for creativity, ambition, and correctness/completeness of implementation, possibly giving extra credit for particularly exciting ideas. Creative solutions would be ones not necessarily present in Facebook or other currently existing popular sites. "Ambition" means that you may not get as many points for the ability to print all user names in capital letters instead of lowercase, compared to — say — implementing a full-fledged message system. "Correctness/Completeness" means that if you started on a full-fledged message system, but nothing actually works, the solution would not be as highly judged as one that actually does what you want it to do.