CS 130A (Fall 2015)

Project - Part 3

Date Assigned: Sunday, November 1, 2015
Due: Monday, November 16, 2015 at 5:00 PM
Submission name for this part: Project_Part3

This is the third part of your longterm Social Network project. You should be building on top of your code for parts 1 and 2. You may also want to fix some things you did back then that you would like to improve. (If you feel it is necessary, you can also start from scratch.)
You may now replace your List and similar classes with suitable C++ STL versions. While doing so, we suggest you replace one occurrence at a time and make sure it works as expected before moving on to the next occurrence. Otherwise, you may find so many problems at once that you feel overwhelmed.
In this and future part, we will specify where you are allowed to use STL classes, and where you should use your own implementations.

1. Review chapter 7 from the textbook.

2. As mentioned at the end of previous part, you may now want to add a functionality for a user to post on a friend's wall. You should have following functionalities in your implementation:
   A user should be able to:

   (a) create a new post on a friend's wall.
   (b) delete his/her own post on a friend's wall.
   (c) delete any friend's post on his/her wall.

   If you haven't done it already, you will need to add a field for storing the author of the post in your WallPost class. You may also want to add fields to store the user whose wall has this post and the time when the post was made.

3. Users should be able to post responses to the posts on their own and their friends' walls. User should be able to delete his/her own responses and not anybody else's. User's wall should now also display the responses along with the posts (including the name of the author and time when the response was made). Responses should also be saved along with the original post (while saving it to the file), so they are available when the program is restarted.
   You probably want to first create a class WallpostResponse (or a similar name), with suitable fields and methods. Then, you should modify your WallPost class to also store a list (or vector or similar) of WallpostResponse.

4. The social network we have formed can be seen as a graph of connected users (friends). The degree of separation between two users is the length of the shortest path connecting them.

   (a) You need to implement the functionality to find out degree of separation between two users in your social network. A logged in user can enter username of another user in the network to find 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 it 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 userselected number.

(b) Implement the functionality that for each user u, finds all the users v such that the degree of separation between u and v is equal to 3. A logged in user u can call this function, then the social networking site will find all users v such that the shortest path between u and v has length 3. If no such user exists, then tit should be reported instead.

5. Generate at least 10000 users with around 100 friends each and some posts each for your social network. If you are truly masochistic, you may do this by hand. Otherwise, we recommend writing a little program (that uses some of the classes and functions from your project) to generate them. It is ok if your users are named sdf3bjh hfvk5kn and similar things. (Though if you prefer, you can also use lists of names to generate more realistic ones.) Make sure that your social network's functions (such as reading in all users, searching for friends, etc.) all still work correctly at this scale.