## Machine Problem 2 : Tiny Social Network Service (SNS)

---

### Overview

The objective of this assignment is to exercise your knowledge about Google Protocol Buffers and gRPC by building a tiny SNS, similar in concept with posting and receiving status updates on Facebook or Twitter. In this service, the following must be considered:

1. Each user(client) has his/her own 'Timeline'. The user is allowed to post to his/her own timeline. That is, whenever a new user appears, the server needs to create a new timeline for the new user.
2. A user is allowed to follow any other user after submitting the **FOLLOW** command. For example, when a user posts an update to his/her own timeline, followers can see it on their own timelines.
3. All timeline must be persistent, i.e. their contents must be stored in files in the local file system. The files must not be encrypted, for now.
4. A user operates in two modes:
    - command mode
    - timeline mode

   When the client program starts, it automatically starts in command mode. The commands the client can send are: **FOLLOW**, **UNFOLLOW**, **LIST**, **TIMELINE**
5. The **LIST** command retrieves from the server the list of existing users and the list of users who follow the current user.
6. The **TIMELINE** command switches a user from command mode to timeline mode and allows the user to post some updates to his/her own timeline, which would then appear on his/her follower's timelines too.
7. Once a user issues the **TIMELINE** command, it can not return to the command mode, and it will stay only in timeline mode.
8. The **UNFOLLOW <*username*>** command removes the current user from the given user's timeline.
9. The **FOLLOW <*username*>** command adds the current user to the given user's timeline.
10. After submitting the **FOLLOW** command, the user sees only new posts. That is, the user can not see any posts that were posted before the user started to follow.
11. After submitting the **TIMELINE** command, the user sees the last 20 posts as follows: "username", "posting time" and "actual posting". The posts are expected to appear in the decreasing order of posting time i.e., the most recent post appears first.
12. All communications will be using Google Protocol Buffers v3 and gRPC. It is expected that the client doesn't wait to pass an input or press the enter key to receive and print a new post in the timeline.
13. The client and server must be started in the following ways:
    - **Server:** ./tsd -p <port>
    - **Client:** ./tsc -h <hostname> -p <port> -u <username>

**Key Points**

The running system will consist of the tiny SNS server (tsd) and the tiny SNS client (tsc). As an environment, you could use the AWS Cloud 9 platform where Google Protocol Buffers v3 and gRPC will have to be installed, and develop the program in C++. (See the document "***Programming Guide HW2***" and "***Environment Setup***")

**What To Hand In**

- **Design**
  Before you start hacking away, write a design document. The result should be a system level design document, which you hand in along with the source code. Do not get carried away with it, but make sure it convinces the reader that you know how to attack the problem. List and describe the components of the system: Client/Server Program, and their interaction.

- **Source Code**
  Hand in the source code, consisting of a makefile, a file tsd.cc, a file tsc.cc and any auxiliary files (for example Google protocol buffers and gRPC files). The code should be easy to read (read: well-commented!). For this assignment, we will not stress test your code. As long as the code executes correctly for all the sample test cases, you will likely receive full credit. The instructors reserve the right to deduct points for code that they consider undecipherable. The code is expected to be submitted using Github. Please follow the instructions below

- **Github**
  Instructions associated with Github submission:
  1. If you haven't created a Github account, it is advised that you create one. It is preferred if TAMU Github is not used as it interferes with the repository pull mechanism for grading.
  2. Create a private repository that will contain all machine problem submissions for CSCE 438
  3. Please add the TA and grader as collaborators so that they have access to the repository for the purpose of grading submissions :
     a. cerodav
     b. Harshithahk
  4. For every machine problem, create a separate subdirectory under the CSCE 438 repository. Please ensure that the sub directory follows the naming pattern **'MP_{machine_problem_number}'**. For e.g. for machine problem 2, this sub directory would be named as **MP_2**

- **How to submit ?**

  Please refer to the Canvas MP2 assignment page. A google form link will be posted there and every student will have to fill all the requested information. Apart from this, make sure that the design documentation is uploaded via Canvas assignment submission system.

**Grading Criteria**

- **20%** Complete design documentation
- **20%** Successful compilation
- **60%** Successful completion of 6 test cases. Refer to the attached sample test cases which cover most scenarios. The ones used for grading would be slightly different.

**References**

- **An Introduction to gRPC and Protocol Buffers for Beginners**
- **grpc.io**