COP5615 – Fall 2019
Project 4.1 – Twitter Engine

Alin Dobra

- **Due Date**: 26 November (till 11:59pm)
- One Submission Per Group
- Submit using eLearning
- **What to Include**:
  - README file including group members
  - Lastnames.zip: the code for the project
  - Lastnames-bonus.zip: the code for the project

1 Problem definition

The goal of this (and the next project) is to implement a Twitter-like engine and (in part 2) pair up with Web Sockets to provide full functionality.

Specific things you have to do are:

In part I, implement the following functionalities:

1. Register account and delete account
2. Send tweet. Tweets can have hashtags (e.g. #COP5615isgreat) and mentions (@bestuser). You can use predefines categories of messages for hashtags.
3. Subscribe to user's tweets.
4. Re-tweets (so that your subscribers get an interesting tweet you got by other means).
5. Allow querying tweets subscribed to, tweets with specific hashtags, tweets in which the user is mentioned (my mentions).
6. If the user is connected, deliver the above types of tweets live (without querying).

Other considerations:

The client part (send/receive tweets) and the engine (distribute tweets) have to be in separate processes. Preferably, you use multiple independent client processes that simulate thousands of clients and a single-engine process.

1. You need to measure various aspects of your simulator and report performance.
2. Write test cases using the elixir's built-in ExUnit test framework verifying the correctness for each task. Specifically, you need to write unit tests and functional tests (simple scenarios in which a tweet is sent, the user is mentioned or re-tweets). Write 2-3 tests for each functionality.

When you submit the project, make sure you include a README that explains what functionality is implemented, how to run the tests, etc. You need to submit a report with performance analysis.

2 Requirements

Input: The input provided (as command line to your program will be of the form:

mix run proj4   num_user   num_msg

Where *num_user* is the number of actors you have to create and *num_msg* is the number of tweets a user has to make.

We can check for project using test cases that you will make. You don't have to print certain output. You can print whatever you want.
Make sure your test cases run using mix test.

Actor modeling: In this project you have to use exclusively the actor facility (GenServer) in Elixir (projects that do not use multiple actors or use any other form of parallelism will receive no credit).

README File: In the README file you have to include the following material:

- Team members
- What is working
- Mention all Functionalities that you implemented
- Mention all the test cases that you created

3 BONUS

For bonus-

1. Simulate periods of live connection and disconnection for users.
2. Simulate a Zipf distribution on the number of subscribers. For accounts with a lot of subscribers, increase the number of tweets. Make some of these messages' re-tweets.

You can also implement some extra functionalities that you want to implement.

Write a **Report-bonus.pdf** to explain your functionalities and submit project4-bonus.zip with your code.

For **Project 4.2**, implement a web interface using phoenix and capture various metrics and send them via Phoenix to the browser.

More Project 4.2 details will be posted soon.