Assignment 1 (80 Points)

(30 points) Consider some client-server architecture as follows. Several clients are registered to the server. Periodically, each client sends message to the server. Upon receiving a message, the server flips a coin and decides to either forward the message to all other registered clients (excluding the original sender of the message) or drops the message altogether. To solve this question, you will do the following:

1. Simulate the behaviour of both the server and the registered clients via GO routines. **(10 points)**
2. Use Lamport's logical clock to determine a total order of all the messages received at all the registered clients. Subsequently, present (i.e., print) this order for all registered clients to know the order in which the messages should be read. **(10 points)**
3. Use Vector clock to redo the assignment. Implement the detection of causality violation and print any such detected causality violation. **(10 points)**

For all the points above, you should try your solution with at least 10 clients.

(50 points) Use Bully algorithm to implement a working version of replica synchronization. You may assume that each replica maintains some data structure (that may diverge for arbitrary reasons), which are periodically synchronized with the coordinator. The coordinator initiates the synchronization process by sending message to all other machines. Upon receiving the message from the coordinator, each machine updates its local version of the data structure with the coordinator's version. The coordinator, being an arbitrary machine in the network, is subject to fault. Thus, a new coordinator is chosen by the Bully algorithm. You can assume a fixed timeout to simulate the behaviour of detecting a fault. The objective is to have a consensus across all machines (simulated by GO routines) in terms of the newly elected coordinator.

1. Implement the above protocol of joint synchronization and election via GO **(20 points)**
2. While implementing your solution, try to simulate both the worst-case and the best-case situation (as discussed in class). **(10 points)**
3. Consider the case where a GO routine fails during the election, yet the routine was alive when the election started. **(5 points + 5 points)**
    a. The newly elected coordinator fails while announcing that it has won the election to all nodes. Thus, some nodes received the information about new coordinator, but some others did not.
    b. The failed node is not the newly elected coordinator.
4. Multiple GO routines start the election process simultaneously. **(5 points)**
5. An arbitrary node silently leaves the network (the departed node can be the coordinator or non-coordinator). **(5 points)**

Assignment Submission Instruction:

Please follow the submission instructions carefully. Note that we need to run your code and need to interpret the outputs generated by your code. Thus, it is important to strictly follow the submission instruction as follows:

1. Collect all submission files and compress them in a zip file. Name the submission of your homework as **PSET1_<your_student_id>.zip**
2. Include a README file in your submission that clearly explains how to compile and run your GO programs. As the homework demands different configurations in

simulation, explain in your README clearly to distinguish the cases (for example, if you have any command line features or arguments, explain them clearly).

3. If you have used any external package for GO (should not be required for this homework), kindly explain them in your README. Note that you are still required to code the protocols yourself.
4. Kindly include some information in the README on how to interpret the output of the program. Note that we will also check the source code. Thus, even though it is not required to excessively comment, a comment per GO routine is appreciated. Basically, each GO routine can carry a comment on what it is supposed to do.
5. Your README should specify how to activate each scenario in line with the homework questions. For example, the question on the coordinator election has five different scenarios. It is OK, if you must handle them using different functions. However, clearly mention in the README about the flags that we need to pass (via the command line) to activate each scenario for evaluation.
6. Any other information that you think helpful for running your code (e.g., open issues, assumptions) will be appreciated too.