Assignment 1 (60 Points)

(30 points) Consider some client-server architecture as follows. A number of clients are registered to the server. Periodically, each client sends message to the server. Once the server receives a message, it distributes the message to all other registered clients. Additionally, the server randomly delays each received message while broadcasting to a registered client.

To solve this question, you will do the following:

1. Simulate the behavior 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. Once the vector clocks are assigned for all the events, check for any causality violation and present the causality violation to the programmer (if any). **(10 points)**

(30 points) Use GO language to implement the original Bully algorithm. You can assume a fixed timeout to simulate the behavior of detecting a fault. You can also randomly select a GO routine to be the victim (i.e., faulty node). The objective is to have a consensus across all GO nodes in terms of the newly elected coordinator.

1. While implementing your solution, try to simulate both the worst-case and the best-case situation (as discussed in class). **(10 points)**
2. 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.
3. Multiple GO routines start the election process simultaneously. **(5 points)**
4. An arbitrary and fresh node (i.e., other existing nodes are not aware of this new node) joins the network while no election was taking place. **(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. Any other information that you think helpful for running your code (e.g., open issues) will be appreciated too.