Assignment 2 (40 marks)

Using the GO language, implement the following distributed mutual exclusion protocols using at least eleven (11) nodes:

1. **(10 marks)** Lamport's shared priority queue with Ricart and Agrawala's optimization.
2. **(10 marks)** Voting protocol with deadlock handling.
3. **(5 marks)** Centralized server protocol without any election.

(15 marks) Compare the performance of the three implementations (in terms of time). In particular, increase the number of nodes simultaneously requesting to enter the critical sections to investigate the performance trend for each of the protocol. For each experiment and for each protocol, compute the time between the first request and all the requesters to exit the critical section. If you have eleven nodes, therefore, your performance table should contain a total of 33 entries (11 entries for each of the three protocols).

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 **PSET2_<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.