EXERCISE 2: DISTRIBUTED MUTUAL EXCLUSION

Learning Objectives
By the end of this exercise, you should be able to:

- Implemented clock based mutexes.
- Understand distributed system synchronization challenges.

Exercise statement
We need to design and implement a distributed application. This application must have two *heavyweight* processes **ProcessA** and **ProcessB**. ProcessA must invoke 3 lightweight processes **ProcessLWA1**, **ProcessLWA2** and **ProcessLWA3**. ProcessB, on the other hand, must invoke 3 processes **ProcessLWB1**, **ProcessLWB2**, **ProcessLWB3**. Each lightweight process must live in a infinity loop which will consist of showing your ID on the screen 10 times and while waiting 1 second.

Both heavyweight processes have to run on the same machine, so all lightweight processes will compete for the same shared resource: the screen. A token-based mutual exclusion policy will need to be implemented between the two heavyweight processes. Among the processes invoked by ProcessA, a *Lamport's policy* must be implemented for mutual exclusion. Among the processes invoked by ProcessB, *Ricart and Agrawala's policy* will need to be implemented for mutual exclusion.

So, when running our distributed application, an output like the following should appear:

```
I'm lightweight process A1
I'm lightweight process A1
...
I'm lightweight process A1
I'm lightweight process A1
I'm lightweight process A2
I'm lightweight process A2
...
I'm lightweight process A2
I'm lightweight process A2
I'm lightweight process A3
I'm lightweight process A3
...
```

```
I'm lightweight process A3
I'm lightweight process A3
I'm lightweight process B1
I'm lightweight process B1
...
I'm lightweight process B1
I'm lightweight process B1
I'm lightweight process B2
I'm lightweight process B2
...
I'm lightweight process B2
I'm lightweight process B2
I'm lightweight process B3
I'm lightweight process B3
...
```

Although all *heavyweight* and *lightweight* processes must run on the same machine, these must be independent programs that can only communicate with each other via *sockets*.

Below is an example of the skeleton of each of the heavyweight processes:

```
...
while(1){
    while(!token) listenHeavyweight();
    for (int i=0; i<NUM_LIGHTWEIGHTS; i++) {
        sendActionToLightweight();
    }
    while(answersfromLightweigth < NUM_LIGHTWEIGHTS) {
        listenLightweight();
    }
    token=0;
    sendTokenToHeavyweight();
}
...
```

Below is an example of the skeleton of each of the lightweight processes:

```
...
while(1){
    waitHeavyWeight();
    requestCS();
    for (int i=0; i<10; i++){
        printf("I'm lightweight process %s\n", myID);
        waitSecond();
    }
    releaseCS();
    notifyHeavyWeight();
}
...
```