TP2 MVFA: Peterson mutual exclusion and DKR leader election

1. Implement the Peterson exclusion algorithm for two processes with NuSMV. You can find information about this algorithm here: `https://fr.wikipedia.org/wiki/Algorithme_de_Peterson` (French) or here: `https://en.wikipedia.org/wiki/Peterson%27s_algorithm` (English). You can use a `pc` variable like we did last time.

   (a) Verify the mutual exclusion property: the two processes do not enter their critical sections at the same time.

   (b) Verify the progress property: both processes can enter their critical sections infinitely often.

   (c) Modify your model to allow a process to stay in the idle state (the state before assigning flag and turn variables), and in the critical section an arbitrary number of steps. Do the properties still hold?

2. The distributed algorithm of Dolev, Klawe and Rodeh (1982) implements leader election in a unidirectional ring. Each process starts with its own identity, which is an integer identifying it in the ring. The goal is that one single process declares itself as the leader. It is also required that each process executes the same algorithm. The DKR algorithm is the following.

   - Each process may work in 2 different modes: while it may still hope being elected as the leader, a process is in *active* mode. As soon as it detects that it will not be elected, it becomes *passive*. In this mode, it just forwards the received messages but cannot become leader anymore.
   - All processes are initially active, and their identity variables are pairwise disjoint but initialized nondeterministically.
   - Each process maintains a variable, called leader, which initially is its own identity. At each step, process are all executed synchronously and do the following actions:
     - transmit the value `val` of the leader variable to its clockwise neighbor in the ring.
     - in active mode, if the received value is equal to the current value of the leader variable, then become leader.
     - in passive mode, set the leader variable to the max of the current value and the received value.

   Model this protocol with NuSMV (you can limit yourselves to 6 processes) and check that there cannot be two processes elected as leaders; and that there exists at least one execution where a leader is selected. You can download `TP2.smv` for an implementation of an unidirectional ring.