Epreuve écrite ASPD

Exercice 1 :

```
------------------------------ MODULE exo1 ------------------------------
EXTENDS Integers,TLC,Sequences
CONSTANTS nbProcess
Pn == 1..nbProcess

(*
--algorithm work1
    variables R = [Rto \in Pn |-> FALSE] ; MessageQueue = <<>>; Done = [Rto \in Pn
|-> FALSE]

    fair process Q = nbProcess+1
    variables tmp;
    begin Qp:
        while true  do
            TraiteMessage:
                receive(mes,mes = "Can I Use R?",Pn);
            SendR:
                send(R,Pn)
            WaitForTermination:
                receive(mes,mes = "My work is done",Q);
        end while;
    end process

    process P \in Pn
    begin
        SendMsg:
            send("Can I Use R ?",Qp)
        Wait:
            receive(mes,true,Q)
        DoSomething:
            DoSomethingWithR(mes)
        bringBackR:
            send("My work is done",Qp)
    end process


end algorithm;
*)
```

Exercice 2 :

Avec n le nombre de processus.

Question 1 : **l'algorithme de Lamport.**

Dans un système à n processus , pi envoie (n−1) messages pour la demande et la libération de la section critique.

Question 2 : **Algorithme Ricart-Agrawala**

Il a pour but de diminuer le nombre de messages échangés par entrée en section critique et élimine les messages de type libération.

Deux types de message sont utilisés ici :

Les messages REQUETE qui sont envoyés lorsqu'un site veut entrer en section critique

Les messages REPONSE qui sont envoyés soit immédiatement à la réception d'un message de type REQUETE, soit ultérieurement à la sortie de section critique du site.

(n−1) messages de type Ok pour rentrer en section critique

Question 3 : **Carvahlo et Roucairol**

C'est le même que l'algorithme Ricart et Agrawala. Il est optimisé à un moment donné : une fois qu'un site reçoit un message de réponse d'un autre site, le premier site peut entrer dans la section critique sans obtenir l'autorisation de l'autre site jusqu'à ce qu'il envoie un e-mail de réponse à l'autre partie.

0 et (2(n−1))

Question 4 :

Dans l'exercice 1, le nombre de message : n (on a un processus donc au bout de 1 message).

Exercice 3 :

Question 1 :

```
---------------- MODULE exo1   --------------------
EXTENDS Naturals,TLC
CONSTANTS ROOTFORCE

VARIABLES
        nb,sn,bm,bt,ba,root,msg,ack,tr,cnt,con
-----------------------------------------------------
NODES == {0,1,2,3,4,5,6}

election(i) ==
    /\ i \in NODES
    /\ nb[i]=sn[i]
        /\ root'=[root EXCEPT![i]= TRUE]
        /\ UNCHANGED <<nb,sn,bm,bt,ba,msg,ack,tr,cnt,con>>
-----------------------------------------------------


sending_msg(x,y) ==
    /\ x \notin bm
    /\ y \notin ba[x]
    /\ nb[x]=sn[x] \cup {y}
        /\ msg' = msg \cup {<<x,y>>}
    /\ bm' = bm \cup {x}
        /\ UNCHANGED <<nb,sn,bt,ba,root,ack,tr,cnt,con>>
-----------------------------------------------------
sending_ack(x,y) ==
    /\ <<x,y>> \in msg
    /\ x \notin ba[y]
    /\ y \notin bm
        /\ ba'=[ba EXCEPT![y]= @ \cup {x}]
    /\ ack' = ack \cup {<<x,y>>}
        /\ UNCHANGED <<nb,sn,bm,bt,root,msg,tr,cnt,con>>
-----------------------------------------------------

progress(x,y) ==
    /\ <<x,y>> \in ack
    /\ x \notin bt
        /\ tr'=tr \cup {<<x,y>>}
    /\ bt' = bt \cup {x}
        /\ UNCHANGED <<nb,sn,bm,ba,root,msg,ack,cnt,con>>


rcv_cnf(x,y) ==
    /\ <<x,y>> \in tr
    /\ x \notin sn[y]
        /\ sn'=[sn EXCEPT![y]= @ \cup {x}]
        /\ UNCHANGED <<nb,bm,bt,ba,root,msg,ack,tr,cnt,con>>


decontention(x,y) ==
```

```
    /\ <<x,y>> \in cnt
    /\  <<y,x>> \in cnt
        /\ msg'=msg - cnt
        /\ bm'= bm - {x,y}
        /\ cnt'= {}
        /\ UNCHANGED <<nb,sn,bt,ba,root,ack,tr,con>>


contention(x,y) ==
    /\ con = 0
    /\ <<x,y>> \in msg
    /\ <<x,y>> \notin ack
    /\ x \notin ba[y]
    /\ y \in bm
        /\ cnt'=cnt\cup {<<x,y>>}
    /\ con'= 1
        /\ UNCHANGED <<nb,sn,bm,bt,ba,root,msg,ack,tr>>

solvecon(x,y) ==
    /\ con = 1
    /\ <<x,y>> \in msg
    /\ x \notin ba[y]
    /\ y \in bm
        /\ ba'=[ba EXCEPT![y]= @ \cup {x}]
    /\ ack' = ack \cup {<<x,y>>}
        /\ UNCHANGED <<nb,sn,bm,bt,root,msg,tr,cnt,con>>


----------------------------------------------------------
Init ==
    /\ nb = [i \in NODES  |-> IF i=0 THEN {1} ELSE IF i = 1 THEN  {0,2,3} ELSE IF
i = 2 THEN  {1,4} ELSE IF i = 3 THEN  {1} ELSE IF i = 4 THEN  {2,6,5} ELSE IF i =
5 THEN  {4} ELSE {4}]
    /\ sn = [i \in NODES  |-> {}]
    /\ bm ={}
    /\ bt ={}
    /\ ack ={}
    /\ ba = [i \in NODES  |-> {}]
    /\ root = [i \in NODES  |-> FALSE]
    /\ msg ={}
    /\ cnt ={}
    /\ tr ={}
    /\ con = 0
Next ==
\/ \E i \in NODES: election(i)
\/ \E x,y \in NODES: sending_msg(x,y)
\/ \E x,y \in NODES: sending_ack(x,y)
\/ \E x,y \in NODES: progress(x,y)
\/ \E x,y \in NODES: rcv_cnf(x,y)
\/ \E x,y \in NODES: contention(x,y)
\/ \E x,y \in NODES: decontention(x,y)
\/ \E x,y \in NODES: solvecon(x,y)
Un_leader_est_elu == <>(\E i \in NODES : root[i] = TRUE)

==========================================================
```

Question 2 :

On ajoute cette ligne pour vérifier l'élection d'un leader.

Effectivement en vérifiant a l'aide TLa, il est possible d'élire un leader

Question 3 :

$$tr \in ND \nrightarrow ND$$