EXAMEN | Théorie et Pratique de la Concurrence | 25/05/2016 15^{30}-18^{00}

Les seuls documents autorisés sont deux feuilles manuscrites A4 recto-verso.
Ce sujet comporte quatre exercices.

Exercice 1 *Algorithme d'exclusion mutuelle pour deux processus*

```
int A=0;      // variables partagées
int B=0;
```

```
-- Processus P
    while(true){
p1:  A=1;
p2:  while(B!=0){}
p3:  Section Critique
p4:  A=0;
    }
```

```
-- Processus Q
    while(true){
q1:  while(A!=0){}
q2:  B=1;
q3:  Section Critique
q4:  B=0;
    }
```

FIGURE 1 – Algorithme d'exclusion mutuelle

On considère l'algorithme d'exclusion mutuelle pour deux processus proposé à la Figure 1 (la section non-critique a été volontairement oubliée pour faciliter le raisonnement).

1. Dessinez le début du diagramme d'états correspondant à cet algorithme (maximum 10 états).
2. Cet algorithme respecte-t-il l'exclusion mutuelle ? Justifiez votre réponse.
3. Cet algorithme garantit-il l'absence d'interblocage ? Justifiez votre réponse.
4. Cet algorithme respecte-t-il l'absence de famine ? Justifiez votre réponse.

Exercice 2 *Analyse d'algorithmes avec des sémaphores*

On considère l'algorithme présenté à la Figure 2 pour 4 processus avec la condition suivante sur les différentes sections critiques : si un processus est en SC0 aucun autre processus n'est en SC0,SC1,SC2, en revanche on peut avoir un processus en SC1 et un en SC2 en parallèle.

1. Cet algorithme garantit-il la propriété sur les sections critiques ? Justifiez votre réponse.
2. Cet algorithme garantit-il l'absence d'interblocage ? Justifiez votre réponse.
3. Cet algorithme garantit-il l'absence de famine ? Justifiez votre réponse.
4. Si une ou plusieurs des propriétés ci-dessus n'est pas satisfaite, existe-t-il une méthode pour corriger le ou les problèmes ? Justifiez votre réponse.
5. On veut rajouter un processes P5 avec une section critique SC3 qui ne peut pas être exécutée en même temps que SC0, ni en même temps que SC1 mais peut-être exécutée en même temps que SC2. Donnez le nouveau code pour les 5 processus de façon à satisfaire la première propriété sur les sections critiques et celle que l'on vient de rajouter. Il est possible de rajouter des sémaphores.

```
    Semaphore S1=(1,[]);      // variables partagées
    Semaphore S2=(1,[]);

-- Processus P0
    while(true){
p01:  Wait(S1);
p02:  Wait(S2);
p03:  SC0
p04:  Signal(S2);
p05:  Signal(S1);
    }

-- Processus P1
    while(true){
p11:  Wait(S1);
p12:  SC1
p13:  Signal(S1);
    }

-- Processus P2
    while(true){
p21:  Wait(S2);
p22:  Wait(S1);
p23:  SC0
p24:  Signal(S2);
p25:  Signal(S1);
    }

-- Processus P3
    while(true){
p31:  Wait(S2);
p32:  SC2
p33:  Signal(S2);
    }
```

FIGURE 2 – Algorithme d'exclusion mutuelle

Exercice 3 *Producteurs et consommateurs avec récipients ordonnés*

On considère un système fait de deux récipients A et B et des producteurs et consommateurs. Le récipient A est de taille a et le récipient B est de taille b où a et b sont des entiers strictement positifs. Les producteurs font en boucle les tâches suivantes, si A et B sont pleins, ils attendent qu'il y ait de la place, si A est plein mais B n'est pas plein ils ajoutent un élément à B et sinon ils ajoutent un élément à A. Quant aux consommateurs, si A et B sont vides ils attendent, si B n'est pas vide il retire un élément de B et sinon il retire un élément de A. Proposez en Java, en C ou en pseudo code , une modélisation de ce systeme en prenant compte que l'on souhaite un processus par_producteur et un processus par consommateur et que l'on veut éviter les attentes actives de ces entités. de plus il vous est demandé de faire attention à la manipulation des variables partagées. On pourra supposer que les récipients sont modélisés par des variables entières partagées.

Exercice 4 *Algorithme par autorisation*

On considère l'algorithme par passage de messages proposé sur la Figure 3 (voir page 3). Le but de cet algorithme est de garantir l'exclusion mutuelle pour 2 processus.

1. Décrivez en quelques lignes le principe de fonctionnement de cet algorithme d'exclusion mutuelle ?
2. Cet algorithme garantit-il l'exclusion mutuelle ? Justifiez votre réponse
3. Cet algorithme garantit-il l'absence de famine ? Justifiez votre réponse et dans le cas d'une réponse négative que faudrait-il garantir pour avoir l'absence de famine ?
4. Comment modifier cet algorithme pour avoir un algorithme d'exclusion mutuelle pour 3 processus (en travaillant éventuellement avec l'hypothèse de la question précédente) ?

```
-- Processus Pi pour i valant 0 ou 1
    boolean B:=true; //variable de P0
    boolean B:=false; //variable de P1
    while(true){
p1:   Section Non Critique
p2:   while(B==false){}
p3:   Section Critique
p4:   B=false;
p5:   send(ok)->P(1-i);
    }

-- Gestion des messages de Pi pour i valant 0 ou 1
case (ok)
    [B=true;]
```

FIGURE 3 – Code du processus i