EXAMEN — Théorie et Pratique de la Concurrence — 26/05/2015 08^{30}-10^{30}

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 V=0;      // variable partagée

-- Processus P                          -- Processus Q
    int lp=0;                              int lq=0;
    while(true){                           while(true){
p1:  Section Non Critique            q1:  Section Non Critique
p2:  V=a;                            q2:  V=b;
p3:  lp=V;                           q3:  lq=V;
p4:  while(lp>0){                    q4:  while(lq>0){
p5:    lp=lp-a;}                     q5:    lq=lq-b;}
p6:  if(lp<0){goto p2;}              q6:  if(lq<0){goto q2;}
p7:  Section Critique                q7:  Section Critique
p8:  lp=0;                           q8:  lq=0;
    }                                      }
```

FIGURE 1 – Algorithme d'exclusion mutuelle

On considère l'algorithme d'exclusion mutuelle pour deux processus proposé à la Figure 1 dans lequel les valeurs entières `a` et `b` ne sont pas fixées.

1. Quelle forme ont les états du diagramme d'états correspondant à ce système ? Pour `a` valant 1 et `b` valant 2, dessinez les premiers états du diagramme (il est inutile de dessiner plus de 10 états).
2. Pour `a` valant 1 et `b` valant 2, ce système respecte-t-il l'exclusion mutuelle ? Justifiez votre réponse. Que se passe-t-il si on change la valeur de `b` mais pas celle de `a` ?
3. Si on prend pour valeur de `a` l'entier 2 et pour `b` l'entier 3, l'algorithme respecte-t-il l'exclusion mutuelle ? Justifiez votre réponse. Qu'en est-il des propriétés d'absence d'interblocage et d'absence de famine ?
4. Pouvez-vous caractérisez les valeurs de `a` et `b` pour lesquelles l'exclusion mutuelle est garantie ?

Exercice 2

Encore un algorithme d'exclusion mutuelle

On considère l'algorithme présenté à la Figure 2.

1. Cet algorithme garantit-il l'exclusion mutuelle ? Justifiez votre réponse.
2. Cet algorithme garantit-il l'absence d'interblocage ? Justifiez votre réponse.
3. Que se passe-t-il par rapport aux deux propriétés précédentes si l'on remplace dans la function `myTurn` la ligne `ret=((req[i] <req[1-i]) || req[1-i]==0);`
   par la ligne `ret=((req[i] <= req[1-i]) ||req[1-i]==0);` ?
4. Comment modifier la function `myTurn` pour garantir à la fois l'exclusion mutuelle et l'absence d'interblocage ? Justifiez votre réponse.

```
int d=0; // Variables Partagées
int req[2]={0;0};

boolean myTurn(int i){
  boolean ret=true;
  ret=((req[i] <req[1-i]) || req[1-i]==0);
  return ret;
}
```

```
-- Processus P0
    while(true){
p1:  req[0]=d;
p2:  d=d+1;
p3:  while(myTurn(0)!=true){}
p4:  Section Critique
p5:  req[0]=0;
    }
```

```
-- Processus P1
    while(true){
q1:  req[1]=d;
q2:  d=d+1;
q3:  while(myTurn(1)!=true){}
q4:  Section Critique
q5:  req[1]=0;
    }
```

FIGURE 2 – Algorithme d'exclusion mutuelle

Les mangeurs de chocolat

Exercice 3

On considère le système fait d'un pot de cubes de chocolat pouvant contenir p cubes de chocolat. On a ensuite un chocolatier qui en boucle remplit le pot de chocolats sachant qu'il peut le remplir avec p nouveaux cubes de chocolat que lorsque le pot est vide. Finalement on a plusieurs mangeurs de chocolat qui en boucle se servent d'un cube de chocolat et qui bien entendu ne peuvent pas se servir si le pot est vide. Proposez en Java, en C ou en pseudo-code, une modélisation de ce système en prenant en compte que l'on souhaite un processus pour le chocolatier et un processus par mangeur de chocolat et que l'on veut éviter les attentes actives du chocolatier et des mangeurs de chocolat.

Passage de jetons

Exercice 4

On considère l'algorithme par passage de messages proposé sur la Figure 3. Le but de cet algorithme est de garantir l'exclusion mutuelle pour N processus (numérotés de 0 à N-1) se trouvant sur des sites différents. Dans cet algorithme, la variable `token` est initialisée à `true` pour le processus 0 et à `false` pour les autres processus. On rappelle que dans un algorithme par passage de messages, deux entités se trouvent sur un même site : l'entité principale et l'entité gérant la réception des messages. Dans l'algorithme donné à la Figure 3, les parties de code entre `[...]` s'exécuteront de façon atomique sur un même site (pas forcément sur différents sites).

1. Décrivez en quelques lignes le principe de fonctionnement de cet algorithme d'exclusion mutuelle ?
2. Cet algorithme garantit-il l'absence de famine ? Justifiez votre réponse et dans le cas d'une réponse négative, proposez une éventuelle modification de l'algorithme.
3. Comment modifier cet algorithme pour autoriser non pas un seul processus en section critique mais p<N ?
4. On souhaite maintenant modifier l'algorithme pour autoriser différents comportements. Par la suite, nous considérons un entier p<N. Voilà les caractéristique de ce nouvel algorithme : les processus numérotés de 0 à p-1 peuvent accéder à la section critique en même temps, en ce qui concerne les processus de p+1 à N-1, ils peuvent accéder à la section critique uniquement si aucun autre processus n'y accède. Proposez un algorithme basé sur l'idée de l'algorithme précédent pour garantir ces comportements ?

```
-- Processus i pour i allant de 0 à N-1
  boolean token=false; //Si i est different de 0
  boolean token=true;  //Si i vaut 0
  boolean CS=false;
  while(true){
p1: Section Non Critique
p2: [CS=true;]
p3: while(token==false){
p4:   send(req)->((i+(N-1)) mod N);}
p5: Section Critique
p6: [CS=false;]
  }
```

```
-- Gestion des messages du Processus i
case (req)  :
  [if(CS==false && token==true){
    send(tok)->((i+1) mod N);
    token=false;
  } else if (token==false) {
     send(req)->(i+(N-1)) mod N);
  }]

case (tok) :
  [token=true;]
```

FIGURE 3 – Code du processus i