Ecole supérieure en sciences et technologies de l'informatique et du numérique **2022/2023**
Architecture parallèle et calcul intensif **1ere année cycle de spécialité (CS1)**

Fiche TD N°1

Exercice 1 : Section critique

1) Quelles sont les conditions que doit vérifier une solution du problème de la section critique ?

2) Soient P1 et P2 deux processus s'exécutant en parallèle et accédant à une même section critique. Soit *libre* une variable partagée de type booléen initialisée à vrai. Considérons les codes des deux processus P1 et P2. Cette solution réalise-t-elle l'exclusion mutuelle des deux processus ?

```
/* code de P1 */
while (1) {
<section non critique>
while (!libre) ;
libre = 0;
<section critique>
libre = 1;
}
```

```
/* code de P2 */
while (1) {
<section non critique>
while (!libre) ;
libre = 0;
<section critique>
libre = 1;
}
```

3) On utilise maintenant une variable partagée *qui*, de type entier, initialisée par le numéro d'un processus (1 ou 2). Cette solution réalise-t-elle l'exclusion mutuelle ? Les codes de P1 et P2 sont :

```
/* code de P1 */
while (1) {
<section non critique>
while (qui != 1) ;
qui = 1;
<section critique>
qui = 2;
}
```

```
/* code de P2 */
while (1) {
<section non critique>
while (qui != 2) ;
qui = 2;
<section critique>
qui = 1;
}
```

4) Dans cette troisième tentative on utilise un tableau tbool partagé de deux booléens initialisé à {0,0}. Montrer que cette solution ne réalise pas l'exclusion mutuelle. Les codes de P1 et P2 sont :

```
/* code de P1 */
while (1) {
<section non critique>
while (tbool[1]) ;
tbool[0] = 1;
<section critique>
tbool[0] = 0;
}
```

```
/* code de P2 */
while (1) {
<section non critique>
while (tbool[0]) ;
tbool[1] = 1;
<section critique>
tbool[1] = 0;
}
```

5) Changeons les codes de nos deux processus. Monter qu'il y a interblocage.

```
/* code de P1 */
while (1) {
<section non critique>
tbool[0] = 1;
while (tbool[1]) ;
<section critique>
tbool[0] = 0;
}3
```

```
/* code de P2 */
while (1) {
<section non critique>
tbool[1] = 1;
while (tbool[0]) ;
<section critique>
tbool[1] = 0;
}
```

Cette dernière solution est une combinaison des solutions précédentes. Elle consiste à utiliser un tableau de deux booléens tbool[2] = {0, 0} ; et un entier qui. Les codes de nos deux processus sont les suivants :

```
/* code de P1 */
while (1) {
<section non critique>
tbool[0] = 1;
qui = 0;
while (tbool[1] && qui == 0)
;
<section critique>
tbool[0] = 0;
}
```

```
/* code de P2 */
while (1) {
<section non critique>
tbool[1] = 1;
qui = 1;
while (tbool[0] && qui ==
1) ;
<section critique>
tbool[1] = 0;
}
```

6) Montrer que cette solution est correcte (C'est-à-dire qu'elle réalise l'exclusion mutuelle et ne présente pas de cas d'interblocage.)

Exercice 2 : Interblocage

Pour traiter une image de taille **T unités,** un système composé de **N processeurs** à mémoire partagée, crée N processus. Chaque processus s'exécute sur un processeur et se charge de traiter une partie de l'image. Pour faire son traitement, un processus a besoin d'une unité de mémoire par unité d'image qu'il a à traiter, mais demande de la mémoire unité par unité au fur et à mesure de ses besoins. Si une demande ne peut être satisfaite, le processus demandeur est mis en attente (attente active). La libération des unités de mémoire allouées à un processus aura lieu lorsqu'il aura fini le traitement de toute sa partie. La mémoire disponible pour répondre aux demandes d'allocation des différents processus est de taille **M unités.**

Partie 1 : Algorithme du banquier

Pour éviter les interblocages, le système utilise l'algorithme du banquier. Supposez que N =4, T = 16, M = 8 et qu'à l'état courant les processus P1, P2, P3 et P4 ont respectivement traité 1 sur 3 unités, 1 sur 4 unités, 1 sur 4 unités et 3 sur 5 unités (par exemple, **P1 : 1 sur 3 unités** signifie que le processus P1 est chargé de traiter 3 unités de l'image et a seulement traité 1 unité).

1) Est-il possible d'atteindre une situation d'interblocage, si T≤M ? Si T>M ? Justifiez votre réponse.
2) Vérifiez, en utilisant l'algorithme du banquier, si l'état courant est certain (sûr ou sauf).
3) Le processus P3 demande une unité de mémoire. Doit-on la lui accorder ? répondre à cette question en utilisant l'algorithme du banquier.

Partie 2 : Caractérisation de l'interblocage

Considérez un état courant où chaque processus Pi détient Ci unités et a encore besoin de Ri unités pour poursuivre et terminer le traitement de sa partie.

1) Exprimez en fonction de M, Ci et Ri, i=1 à N, la condition qui caractérise la situation d'interblocage (c'est-à-dire : aucune ressource n'est disponible et il y a encore des besoins).
2) Sachant que $T = \sum_{i=1}^{N}(Ci + Ri)$, montrez que si T < N+M alors il n'y a pas d'interblocage à l'état courant.

Annexe :

L'algorithme du banquier consiste à examiner chaque nouvelle requête pour voir si elle conduit à un état sûr

- Le cas échéant la ressource est allouée, sinon elle est mise en attente
- Afin de voir si un état est sûr :
  - Le banquier vérifie s'il possède suffisamment de ressource pour satisfaire un client
  - Si tel est le cas, on suppose que ces crédits seront remboursé, on examine le cas du client le plus proche de la limite et ainsi de suite.
  - Si tous les crédits sont finalement remboursés, l'état est sûr et la requête initiale peut être raccordée.

Processus	Allocation	Max	Available
	$R_1, R_2,\ldots,R_n$	$R_1, R_2,\ldots,R_n$	$R_1, R_2,\ldots,R_n$
P_1			
P_2			
…			
P_m			