Université KASDI Merbah de Ouargla
Faculté des NTIC/ Département ITI

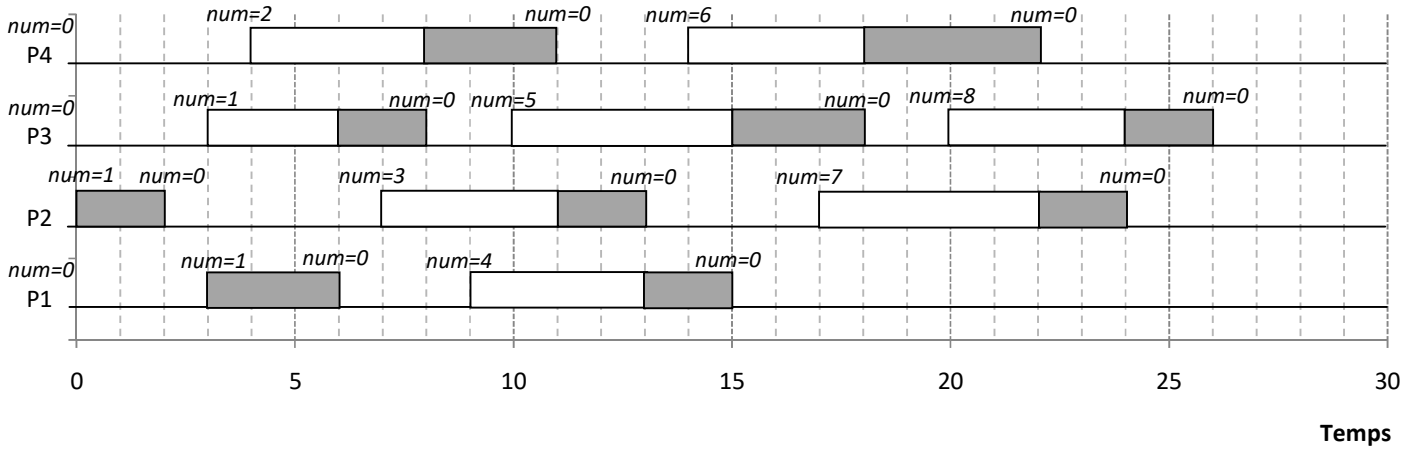
Corrigé type examen ADis (M1 Fond / Ind S2) 1h :30

Exercice 1 (8 pts):

Le tableau suivant décrit le scénario d'exécution de quatre processus P1, P2, P3, P4 :

P1	P2	P3	P4
(3,3) (9,2)	(0,2) (7,3) (13,2)	(3,2) (7,3) (12,2)	(4,3) (10,4)

- Donner le diagramme temporel de l'exécution en appliquant l'exclusion mutuelle par l'algorithme de la Boulangerie et indiquer à chaque fois la mise à jour du variable ***num de Pi*** :



Exercice 2 (7 pts):

- Considérons un système contenant 3 sites. Supposons que l'horloge matricielle local du site 3 est (HM3).
- Le site 3 reçoit successivement deux messages m1 (EMm1) et m2 (EMm2) en provenance du site 2.

$$HM3=\begin{bmatrix}3 & 1 & 1\\ 1 & 5 & (2)\\ 1 & 2 & 6\end{bmatrix} \qquad EMm1=\begin{bmatrix}4 & 2 & 1\\ 2 & 10 & (4)\\ 1 & 2 & 5\end{bmatrix} \qquad EMm2=\begin{bmatrix}4 & 2 & 1\\ 1 & 8 & (3)\\ 1 & 2 & 5\end{bmatrix}$$

- Quel est l'ordre de délivrance des messages m1, m2 avec justification:
  - ✓ *Pour le message m1, on a (EMm1[2,3]=4) ≠(HM3[2,3]+1=3), donc m1 ne respect pas l'ordre FIFO sur le canal C23.*
  - ✓ *Pour le message m2 :*
    1. *EMm2[2,3]=HM3[2,3]+1=3 (vérifié)*
    2. *EMm2[1,3]=HM3[1,3] (≤ vérifié)*

    → le message m2 est délivrable alors m-à-j HM3 → $HM3=\begin{bmatrix}4 & 2 & 1\\ 1 & 8 & 3\\ 1 & 2 & 7\end{bmatrix}$
  - ✓ *Revenons au message m1*
    1. *EMm1[2,3]=HM3[2,3]+1=4 (vérifié)*
    2. *EMm1[1,3]=HM3[1,3] (≤ vérifié)*

    → le message m1 est devenu délivrable alors m-à-j HM3 → $HM3=\begin{bmatrix}4 & 2 & 1\\ 1 & 10 & 4\\ 1 & 2 & 8\end{bmatrix}$

Exercice 3 (5 pts):

Pour implémenter le tableau distribué ***num[]*** sur ***n*** sites ***(S_0,.., S_{n-1})*** utilisé dans l'algorithme de la boulangerie, nous déclarons un variable local ***int num*** . Chaque processus ***Pi*** peut acceder directement à sont variable local ***num***. ***Pi*** peut accéder au variable ***num*** d'un autre processus ***Pj (j≠i)*** à l'aide de la méthode à appel distant ***int getNum()*** et à travers l'objet distant ***dist[j]*** *:* « ***dist[j].getNum() »***.

- Ecrire le code Java de la méthode ***int getNum()***.

- Ecrire le code Java de la méthode ***boolean minNum(int i)*** qui retourne ***true*** si le couple ***(i, num)*** du processus ***Pi*** ayant la valeur minimale parmi tous les processus ***Pj (j :0..n-1 et j≠i)***.

Remarque : on dit que **(i, numi) < (j,numj) si (numi < numj ou (numi = numj et i<j))**.

```
/**************** Variables à utiliser **************/
/* dist[] : tableau des objets distants de taille n */
/* num : variable local entier */
/****************************************************/
public int getNum(){
   return num ;
}
public boolean minNum(int i){
   for(int j=0 ;j<n ;j++){
     if(j != i)
        if(dist[j].getNum() < num || (dist[j].getNum() == num && j<i))
            return false ;
   }
   return true ;
}
```