TP 3 Systèmes concurrents et distribués

Exercice 1 :

Montrez que l'implémentation des sémaphore binaires vue au cours n'assure pas la vivacité.

Ecrire une implémentation des fonctions P() et V() pour une sémaphore binaire qui assure qu'un thread qui appelle P() entrera en section critique (pas de famine).

Idée : utilisez deux files d'attente. Les threads dans la seconde file d'attente sont servis d'abord. Lorsqu'il n'y a plus de threads dans la seconde file d'attente, tous les threads dans la première passent dans la seconde.

Exercice 2 : L'algorithme de Kessel ci-dessous est une solution du problème d'exclusion mutuelle pour deux processus. Montrez qu'il résout le problème de l'exclusion mutuelle et est starvation free et deadlock free. Utilisez le modèle d'entrelacement et les variables partagées sont toutes **volatiles**. Pour rappel, les lectures écritures d'une variable volatile par deux threads sont ordonnées, c'est-à-dire l'une s'exécute toujours avant l'autre (on a un flèche entre les actions des deux threads).

Initialement : b={false,false}, turn=quelconque

Thread 0

```
b[0]=true;
local[0]=turn[1];
turn[0]=local[0];
while ((b[1]==true && (local[0]==turn[1]))
```

Thread 1

```
b[1]=true;
local[1]=1-turn[0];
turn[1]=local[1];
while ((b[0]==true && (local[1]!=turn[0]))
```

critical section

b[0] = **false**;

critical section

b[1] = **false**;