TD2: SYNCHRONISATION

Nom: Nguyen Sang
Code'etudient: 18126029

Exercice 2:

semaphore mutex, leave, empty;
init(mutex, 1); **init**(leave,N); **init**(empty, 0)

```
Client() {
    if(nbClients < N) {
        nbClients = nbClients + 1;
        up(&empty);
        sefairecoiffer();
        down(&leave);
        sortir();
    }
    else {
        sortir();
    }
}
```

```
Coiffeur(){
    while (true){
        down(&empty);
        down(&mutex);
        nbClients = nbClients - 1;
        up(&mutex);
        coiffer();
        up(&leave);
    }
}
```

Exercice 3:

a)

int nbCars = 0; int wait = 0; int parkingLot = N;

```
barrierE() {
    sePresenterE();
    if(noCarAtGateE){
        nbCars = nbCars + 1;
        while(true){
            if( nbCars < parkingLot ){
                parkingLot-= 1;
                enterCarPark();
                break;
            }
            else{
                wait = 1;
                while( wait == 1 )
                    wating();
            }
        }
    }
    else{
        sortir();
    }
}
```

```
barrierS(){
    while(true){
        sortir();
        nbCars = nbCars - 1;
        wait = 0;
        parkingLot += 1;
        sePresenterS();
    }
}
```

b) Y-a-t-il des problèmes de concurrent ? Donner un exemple de concurrent.

- À mon avis, il y a des problèmes de concurrents comme:
+) Lorsque **deux** voitures entrent dans la porte E, **la première voiture** voit qu'il n'y a pas de voiture à la porte E, puis elle entre dans la porte. Mais à ce moment, **le système d'exploitation** a soudainement **arrêté** ce processus, puis **une autre voiture** voit toujours qu'il n'y a pas de voiture à la porte E, puis elle entre également dans la porte. Les **nbCars** seront augmentés de **1** unité au lieu de **2**.

+) Un autre problème est qu'un processus ne satisfait pas l'expression conditionnelle de "**if** (if(nbCars < N))", elle doit attendre que la voiture de stationnement soit vide. Alors que le processus décide d'attribuer la variable "**wait**" par **1**, **le système d'exploitation** soudainement **arrête** le processus et les autres voitures suivantes viennent et font de même. → En conséquence, de nombreuses voitures sont bloquées dans la section d'attente.

c)

```
barrierE() {
    sePresenterE();
    if(noCarAtGateE){
        nbCars = nbCars + 1;
        while(true){
            if( nbCars < parkingLot ){
                parkingLot-= 1;
                enterCarPark();
                break;
            }
            else{
                wait = 1;
                while( wait == 1 )
                    wating();
            }
        }
    }
    else{
        sortir();
    }
}
```

```
barrierS(){
    while(true){
        sortir();
        nbCars = nbCars -  1;
        wait = 0;
        parkingLot += 1;
        sePresenterS();
    }
}
```

d) Est-ce qu'on peut utiliser les sémaphores pour implémenter ce moniteur ? Si oui, donner le pseudo-code.

semaphore mutex, attendre, empty;
int nbCars = 0;
init(mutex, 1) , **init**(attendre, 0), **init**(empty, N)

```
barrierE () {
    sePresenterE();
    if( noCarAtGateE ){
        down(&empty);
```

```
barrierS();
    while(true){
        up(&empty);
        if(nbCars > 0){
```

```
        down(&mutex);
        nbCars = nbCars + 1;
        up(&mutex);
        if( nbCars  < N){
            enterCarPark();
        }
        else(
            down(&attendre);
            enterCarPark();
        )
    }
    else{
        sortir()
    }
}
```

```
            sortir();
            sePresenterS();
            down(&mutex);
            nbCars = nbCars – 1;
            up(&mutex);
            up(&attendre);
        }
    }
```

Exercise 4:

Semaphore mutex, full, empty;
init(mutex,1); init(full, 4); int(empty,0);
int count = 0;

```
producingGlass(){
    while(true){
        item = producingItem();
        down(&full);
        down(&mutex);
        insertItem(item);
        count++;
        up(&mutex);
        if( count == 4 )
            up(&empty);
    }
}
```

```
packagingGlass(){
    while(true){
        down(&empty);
        down(&mutex);
        packaging(); //mettent en un paquette
        up(&mutex);
        while(count != 0){
            up(&full);
        }
        deliverProduct(); //livrer ce paquette
    }
}
```

Exercice 6:

Supposez un système ayant 3 ressources (R1, R2, R3) et 4 processus (P1 à P4) qui partagent ces ressources. Les nombres disponibles de trois ressources (R1, R2, R3) à l'instant T0 sont 4, 1, 2. Donner l'allocation des ressources à l'instant T0 comme la table ci-après :

	Allocation			Maximal			Besoin		
	C	D	E	C	D	E	C	D	E
P1	1	0	0	3	2	2	2	2	2
P2	2	1	1	6	1	3	4	0	2
P3	2	1	1	3	1	4	1	0	3
P4	0	0	2	4	2	2	4	2	0

P2: C(6) D(2) E(3)
P1: C(7) D(2) E(3)
P3: C(9) D(3) E(4)
P4: C(9) D(3) E(6)
→ Pas de inter-blocages