Exercice 1

Solution1 :

P1

```
Int i, j;
For (i=0; i<n; i++)
  For (j=0; j<n/2; j++)
  Mat3[i,j]=Mat1[i,j]+Mat2[i,j]
```

P2

```
Int i, j ;
For (i=0; i<n; i++)
  For (j=n/2; j<n; j++)
  Mat3[i,j]=Mat1[i,j]+Mat2[i,j]
```

Solution2 : (sémaphores)

Variables globales :

```
int Mat1[n, n], Mat2[n, n], Mat3[n, n], i=0;

const N=100 ; semaphore mutex=1 ;
```

Processus-1

```
int l=0, j ;
while l<N{
P(mutex);
l=i;
i++;
V(mutex)
If(l<N)
For(j=0;j<N;j++)
Mat3(l,j)=Mat1[l,j]+Mat2[l,j];
```

Processus-2

```
int l=0, j ;
while l<N{
P(mutex);
l=i;
i++;
V(mutex)
If(l<N)
For(j=0;j<N;j++)
Mat3(l,j)=Mat1[l,j]+Mat2[l,j] ;
```

Processus-n

```
int l=0, j ;
while l<N{
P(mutex);
l=i;
i++;
V(mutex)
If(l<N)
For(j=0;j<N;j++)
Mat3(l,j)=Mat1[l,j]+Mat2[l,j] ;
```

Exercice 2

Effectuer un rendez-vous entre 3 processus P1, P2 et P3 en utilisant les sémaphores :

Processus-1

```
V(S1)
V(S1)
P(S2)
P(S3)
```

Processus-2

```
V(S2)
V(S2)
P(S1)
P(S3)
```

Processus-3

```
V(S3)
V(S3)
P(S1)
P(S2)
```

Généralisez votre solution pour N processus :

Solution1 :

Variables globales :

```
Int compteur=0, const n= 100 ;
Semaphore mutex=1 ;
Semaphore S=0 ;
```

Processus-1

```
Int i,position;
_
_
_
P(mutex)
compteur++;
position=compteur;
V(mutex)
if(position<n)
P(S);
else
for(i=1;i<n;i++)
V(S)
_
_
_
```

Processus-2

```
Int i,position;
_
_
_
P(mutex)
compteur++;
position=compteur;
V(mutex)
if(position<n)
P(S);
else
for(i=1;i<n;i++)
V(S)
_
_
_
```

Processus-3

```
Int i,position;
_
_
_
P(mutex)
compteur++;
position=compteur;
V(mutex)
if(position<n)
P(S);
else
for(i=1;i<n;i++)
V(S)
_
_
_
```

Processus-n

```
Int i,position;
_
_
_
P(mutex)
compteur++;
position=compteur;
V(mutex)
if(position<n)
P(S);
else
for(i=1;i<n;i++)
V(S)
_
_
_
```

Solution2 :

```
Int compteur=0, const n= 100 ;
Semaphore mutex=1 ;
Semaphore S=0 ;
```

Processus-1

```
Int i;
_
_
_
P(mutex)
compteur++;
if(compteur<n){
V(mutex);
P(S);
}
else{
V(mutex)
for(i=1;i<n;i++)
V(S)
}
_
_
_
```

Processus-2

```
Int i;
_
_
_
P(mutex)
compteur++;
if(compteur<n){
V(mutex);
P(S);
}
else{
V(mutex)
for(i=1;i<n;i++)
V(S)
}
_
_
_
```

Processus-3

```
Int i;
_
_
_
P(mutex)
compteur++;
if(compteur<n){
V(mutex);
P(S);
}
else{
V(mutex)
for(i=1;i<n;i++)
V(S)
}
_
_
_
```

Processus-n

```
Int i;
_
_
_
P(mutex)
compteur++;
if(compteur<n){
V(mutex);
P(S);
}
else{
V(mutex)
for(i=1;i<n;i++)
V(S)
}
_
_
_
```

Exercice 3

Solution1 :

Processus-1

```
_
_
_
V(S1)
P(S2)
_
_
_
_
_
V(S1)
V(S1)
V(S1)
V(S1)
P(S2)
P(S3)
P(S4)
P(S5)
```

Processus-2

```
_
_
_
V(S2)
P(S1)
_
_
_
_
_
V(S2)
V(S2)
V(S2)
V(S2)
P(S1)
P(S3)
P(S4)
P(S5)
```

Processus-3

```
_
_
_
V(S3)
V(S3)
P(S4)
P(S5)
_
_
_
V(S3)
V(S3)
V(S3)
V(S3)
P(S1)
P(S2)
P(S4)
P(S5)
```

Processus-4

```
_
_
_
V(S4)
V(S4)
P(S3)
P(S5)
_
_
_
V(S4)
V(S4)
V(S4)
V(S4)
P(S1)
P(S2)
P(S3)
P(S5)
```

Processus-5

```
_
_
_
V(S5)
V(S5)
P(S3)
P(S4)
_
_
_
V(S5)
V(S5)
V(S5)
V(S5)
P(S1)
P(S2)
P(S3)
P(S4)
```

Solution2 :

Processus-1

```
_
_
_
V(S1)
P(S2)
_
_
_
_
_
V(S1)
P(S5)
```

Processus-2

```
_
_
_
V(S2)
P(S1)
_
_
_
_
_
V(S2)
P(S5)
```

Processus-3

```
_
_
_
V(S3)
V(S3)
P(S4)
P(S5)
_
_
_
V(S3)
P(S5)
```

Processus-4

```
_
_
_
V(S4)
V(S4)
P(S3)
P(S5)
_
_
_
V(S4)
P(S5)
```

Processus-5

```
_
_
_
V(S5)
V(S5)
P(S3)
P(S4)
_
_
_
P(S1)
P(S2)
P(S3)
P(S4)
V(S5)
V(S5)
V(S5)
V(S5)
```

Exercice 4

1) l'algorithme de chaque client.

```
semaphore S = N;
```

Processus Client

```
Début
P(S)    /* prendre un chariot */
Effectuer les achats
V(S)    /* libérer le chariot */
Fin
```

2) algorithmes des processus « abonnés » et « non abonnés »

```
chariot_utilisé : nombre de chariots utilisé = nombre de clients dans le magasin ;
nbr_abonné : nombre de processus abonnés qui attendent ;
na2 nbr_n_abonné : nombre de processus non abonnés qui attendent.

semaphore S1=0, S2=0, mutex=1 ;
int chariot_utilisé=0, nbr_abonné=0, na2=0 ; /* chariot_utilisé=0; nbr_abonné=0; na2=0 */
```

Processus Client_Abonné

```
P(mutex)
si chariot_utilisé < N alors
  chariot_utilisé := chariot_utilisé + 1
  V(mutex)
sinon
  nbr_abonné := nbr_abonné + 1
  V(mutex)
  P(S1)
finsi
Effectuer les achats
P(mutex)
si nbr_abonné > 0 alors
  nbr_abonné := nbr_abonné - 1
  V(S1)
sinon /* nbr_abonné =0 */
  si nbr_n_abonné > 0 alors
    nbr_n_abonné := nbr_n_abonné - 1
    V(S2)
  sinon /* nbr_abonné = nbr_n_abonné =0 */
    chariot_utilisé := chariot_utilisé - 1
  finsi
finsi
V(mutex)
Fin
```

Processus Client_Non_Abonné

```
P(mutex)
si chariot_utilisé < N alors
  chariot_utilisé := chariot_utilisé + 1
  V(mutex)
sinon
  nbr_n_abonné := nbr_n_abonné + 1
  V(mutex)
  P(S2)
finsi
Effectuer les achats
P(mutex)
si nbr_abonné > 0 alors
  nbr_abonné := nbr_abonné - 1
  V(S1)
sinon /* nbr_abonné =0 */
  si nbr_n_abonné > 0 alors
    nbr_n_abonné := nbr_n_abonné - 1
    V(S2)
  sinon /* nbr_abonné = nbr_n_abonné =0 */
    chariot_utilisé := chariot_utilisé - 1
  finsi
finsi
V(mutex)
Fin
```