Problème de la voir unique :
Soit un pont, à voie unique, traversé pas des véhicules en sens inverse (sens A et sens B). A tout moment, le pont ne doit contenir que des véhicules allant dans un sens uniquement.
1. Parmi les modèles étudiés en classe, ce problème correspond à quel modèle ? Modèle Lecteurs avec deux types de lecteurs !

2. Proposer une solution en utilisant les sémaphores (les codes de demandes d'accès et de sorties) de façon à ce que les processus respectent les règles de circulation, si
1) le nombre de véhicules sur le pont est 1 : un simple mutex pour les deux processus suffit pour protéger la section critique (pont)

Shared semaphore mutex=1;

Voiture_AB(){
P(mutex) ;
Circuler_AB() ;
V(mutex) ;
}
	Voiture_BA(){
	P(mutex) ;
	Circuler_BA() ;
	V(mutex) ;
	}

2) le nombre de véhicules le pont dans un sens est illimité : une variable qui compte le nombre de voitures et la première bloque l'accès à l'autre voie, la dernière libère la voie, tout protégé par des mutex ! la file d'attente S ne contient qu'un seul processus bloqué !

Shared semaphore mutex1=1, mutex2=1, S=1; int nb_AB=0, nb_BA=0;

Voiture_AB(){
P(mutex1) ;
nb_AB++ ;
if (nb_AB==1) P(S) ;
V(mutex1);
Circuler_AB() ;
P(mutex1) ;
nb_AB-- ;
if (nb_AB==0) V(S) ;
V(mutex1) ;
}
	Voiture_BA(){
	P(mutex2) ;
	nb_BA++ ;
	if (nb_BA==1) P(S) ;
	V(mutex2);
	Circuler_BA() ;
	P(mutex2) ;
	nb_BA-- ;
	if (nb_BA==0) V(S) ;
	V(mutex2);
	}

3) si le nombre est limité (k voitures sur le pont simultanément) : il suffit d'ajouter un sémaphore SK initialisé à k !
Shared semaphore mutex1=1, mutex2=1, S=1, SK=k; int nb_AB=0, nb_BA=0;

Voiture_AB(){
P(mutex1) ;
nb_AB++ ;
if (nb_AB==1) P(S) ;
V(mutex1);
P(SK);
Circuler_AB() ;
V(SK) ;
P(mutex1) ;
nb_AB-- ;
if (nb_AB==0) V(S) ;
V(mutex1) ;
}
	Voiture_BA(){
	P(mutex2) ;
	nb_BA++ ;
	if (nb_BA==1) P(S) ;
	V(mutex2);
	P(SK);
	Circuler_BA() ;
	V(SK);
	P(mutex2) ;
	nb_BA-- ;
	if (nb_BA==0) V(S) ;
	V(mutex2);
	}

Problème du Carrefour

Les processus voitures (2 types) sont en compétition (comme des rédacteurs dans la première question puis comme des lecteurs dans la 2eme question). L'accès est donné par le 3eme processus Feux. La variable booléenne « a » indique l'accès et les feux change toute les m minutes (par exemple 10).

1) une seule voiture dans le carrefour à la fois : ici un simple mutex1/2 suffit (pour chacun) pour donner l'accès.

Shared semaphore mutex1=1, mutex2=1, feu1=1, feu2=0; Boolean: a=TRUE; int m=10;

Traversée1( ){
P(mutex1);
P(feu1);
Rouler() ;
V(feu1) ;
V(mutex1) ;
}
	Feux( ){While(TRUE){
	Wait(m) ;
	If(a){P(feu1); V(feu2);}
	else{P(feu2);V(feu1);}
	a=¬a;
	}}
		Traversée2( ){
		P(mutex2);
		P(feu2);
		Rouler() ;
		V(feu2) ;
		V(mutex2) ;
		}

2) k voitures à la fois dans le carrefour : mutex1/2 est init à k, et on a besoin d'un sémaphore W pour laisser passer la dernière voiture (il y'en a k) quand les feux change. On a besoin d'un compteur de voiture (protégé par mutex) pour voir quelle est la première et quelle est la dernière.

Remarque: initialement les voiture de type Traversée 1 auront le privilège de passer pendant m minutes, puis l'alternance s'applique !

Shared semaphore mutex1=k, mutex2=k, feu1=1, feu2=0, mutex=1, W=1;
Boolean: a=TRUE; Int m=10, nv=0;

Traversée1( ){
P(mutex1);
P(feu1);
P(mutex) ;
nv++ ;
if(nv==1) P(W) ;
V(mutex);
V(feu1);
Rouler() ;
P(mutex) ;
nv-- ;
if(nv==0) V(W) ;
V(mutex) ;
V(mutex1) ;
}
	Feux( ){While(TRUE){
	Wait(m) ;
	If(a){P(feu1);
	P(W);
	V(feu2);
	V(W);}
	else{P(feu2);
	P(W);
	V(feu1);
	V(W);}
	a=¬a;
	}}
		Traversée2( ){
		P(mutex2);
		P(feu2);
		P(mutex) ;
		nv++ ;
		if(nv==1) P(W) ;
		V(mutex);
		V(feu2);
		Rouler() ;
		P(mutex) ;
		nv-- ;
		if(nv==0) V(W) ;
		V(mutex) ;
		V(mutex2) ;
		}