DS4 - Corrigé type CCINP

Concurrence

Q1. Il peut y avoir des conflits de lecture/écriture.
Par un exemple, un thread lit la valeur de la variable a, le deuxième thread lit cette même valeur, le thread 1 la modifie et le thread 2 écrase cette modificat° sans en tenir compte.

Q2. On attend l'algo de Peterson.

Q3. Il faut initialiser le tableau pas_pris à true.
Avec cette hypothèse, montrons l'invariant.

Avant la boucle: ensemble vide, rien à vérifier.

Inv: Supposons que $\forall id < i-1$, id a terminé la partie 1 du code et considérons le i^e tour de boucle.
Pour sortir du while, on doit lire pas_pris[i] à false et donc le thread d'indice i a fait cette affectation, au moment où cette affectation est faite, il détient le mutex[i] mais pour passer au $(i+1)^e$ tour de boucle il faut l'acquérir ainsi qd on y arrive on est sûr que le thread i a relâché le mutex et donc terminé la partie 1.

Q4. Pour que tout le monde entre en partie 2, la boucle doit être terminée et c'est nécessaire par l'invariant que tous les threads aient terminé la partie 1 pour que ce soit possible.

Q5. Le thread qui est le plus avancé dans la boucle for, disons qu'il est à l'indice i, pourra forcément avancer si on a supposé que tous les threads ont terminé la partie 1.
Si deux threads sont dans le même tour de boucle, au moins l'un des deux pourra acquérir le mutex[i%n]. Il y a donc toujours au moins un thread qui peut avancer et pas d'interblocage.

Q6. On peut considérer le nb total de tours de boucles à effectuer par l'ensemble des threads. La q° précédente garantit qu'il décroit strictement. Il deviendra donc nul et alors tout le monde entre en partie 2. La condition est donc bien suffisante.