+4/1/42+

UNIVERSITÉ NICE SOPHIA ANTIPOLIS — POLYTECH'NICE SOPHIA

Année 2019-20
Programmation concurrente
QCM n°1

Nom et prénom : Montoya Damien

*Vous devez obligatoirement répondre en noicissant les cases sans utiliser le blanc masque. Certaines questions n'ont peut être pas de bonnes réponses, d'autres une ou plusieurs.*

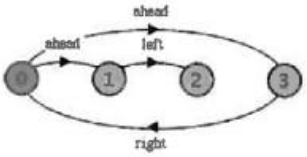
*Barême :*

- *Questions fermées simples :* +3 bonne réponse, 0 pas de réponse, -0.5 mauvaise réponse ou plus d'une case cochée

- *Questions fermées multiples (♣) :* +1 bonne case cochée ou mauvaise case non cochée, 0 pas de réponse, -0.5 bonne case non cochée ou mauvaise case cochée, -1 minimum possible

- *Questions ouvertes :* le barême est indiqué dans le cartouche

**Question 1 ♣** Quel est le processus FSP correspondant à la description LTS suivante (MOVE) :




- [ ] `MOVE=(ahead -> P | ahead -> left). P=(right -> MOVE).`
- [x] `MOVE=(ahead -> right -> MOVE | ahead -> left -> STOP).`
- [x] `MOVE=(ahead -> P | ahead -> Q), P=(right -> MOVE), Q=(left -> STOP).`
- [ ] `MOVE=(ahead -> right -> MOVE | ahead -> left -> MOVE).`
- [ ] *Aucune de ces réponses n'est correcte.*

**Question 2** Quel est le processus ||S qui permet à plus deux processus CLIENTs, d'utiliser le processus SERVEUR ?

```
CLIENT = (appel -> reponse -> traite -> CLIENT).
SERVEUR = (appel-> service -> reponse -> SERVEUR).
```

- [ ] `||S=([1..2]::CLIENT || [1..2]::SERVEUR).`
- [ ] `||S=([1..2]:CLIENT || [1..2]:SERVEUR).`
- [ ] `||S=([1..2]::CLIENT || [1..2]:SERVEUR).`
- [x] `||S=([1..2]:CLIENT || [1..2]::SERVEUR).`

**Question 3** Est-ce que ces deux opérations read et write ci-dessous, regroupées dans la même classe Disk sont correctement programmées ?

```
// Les opérations seek, read et write s'exécutent en exclusion mutuelle
int disk_read (sector x) {
        int r;
        D.seek(x);
        r := D.read();
        return (r);
}

void disk_write (sector x, int v) {
     D.seek(x);
     D.write(v);
}
```

0/3 ☐ oui ☒ non

**Question 4** Est-ce que les processus S1 et S2 ont le même comportement ?

```
P = (a -> b -> P).
Q = (c -> b -> Q).
||S1 = (P || Q).

S2 = (a -> c -> b -> S2 | c -> a -> b -> S2).
```

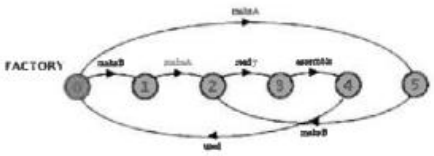
3/3 ☐ non ☒ oui

**Question 5** Soit le programme FSP suivant :

```
MAKE_A   = (makeA->ready->used->MAKE_A).
MAKE_B   = (makeB->ready->used->MAKE_B).
ASSEMBLE = (ready->assemble->used->ASSEMBLE).

||FACTORY = (MAKE_A || MAKE_B || ASSEMBLE).
```

Est-ce que le diagramme LTS suivant correspond au processus ||FACTORY ?




3/3 ☒ oui ☐ non

**Question 6** Soit le programme Java suivant utilisé dans une application :

```
class Compte {
   private int solde = 100;
   public int getSolde()  { return solde; }
   public void retirer (int montant) { solde -= montant; }
   public void deposer (int montant) { solde += montant; }
}
```

On souhaite réutiliser ce code mais maintenant, deux threads souhaitent partager un même objet instance de cette classe. Si vous pensez qu'il faut modifier le code de la classe 'Compte', proposer la nouvelle version de la classe ci-dessous ; sinon laisser la case vide. ☐0 ☐1 ☐2 ☐3 ☐4

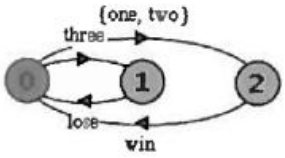
Semaphore retirer = new Semaphore (0);

Semaphore deposer = new Semaphore (0);

Semaphore parage = new Semaphore (1);

Les sémaphores sont déclarés mais pas utilisé. Un seul sémaphore initialisé à 1 est nécessaire pour mettre en oeuvre une section critique

**Question 7** ♣ Quel est le processus FSP correspondant à la description LTS suivante :




- [X] `GAME=({one, two} -> win -> GAME | three -> lose -> GAME).`
- [X] `GAME=(one -> win -> GAME | two -> win -> GAME | three -> lose -> GAME).`
- [ ] `GAME=(one -> P | two -> P | three -> Q). P=(win -> GAME). Q=(lose -> GAME).`
- [X] `GAME=(one -> P | two -> P | three -> Q), P=(win -> GAME), Q=(lose -> GAME).`
- [ ] `GAME=(one -> p | two -> p | three -> q), p=(win -> GAME), q=(lose -> GAME).`
- [ ] *Aucune de ces réponses n'est correcte.*

**Question 8** ♣ Quelle sont les conditions nécessaires pour avoir des données incohérentes :

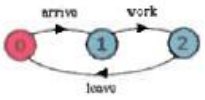
- [ ] Une seule thread
- [ ] Une ressource exclusive
- [X] Plusieurs threads
- [X] Une ressource partagée en lecture et écriture
- [ ] Une ressource partagée en lecture
- [ ] *Aucune de ces réponses n'est correcte.*

**Question 9** Est-ce que le code Java suivant implémente bien l'accès à une section critique ?

```
Semaphore mutex = new Semaphore (0);
mutex.Acquire ();
// je suis en section critique
mutex.Release ();
```

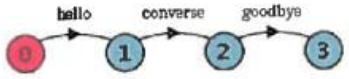
☒ non ☐ oui

**Question 10** ♣ Quel est le processus FSP correspondant à la description LTS suivante :



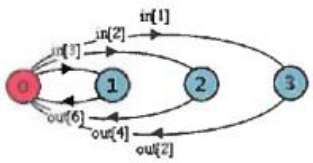

- ☐ `JOB=(arrive -> work -> leave -> job).`
- ☒ `JOB=(arrive -> work -> leave -> JOB).`
- ☐ `JOB=(arrive | work | leave | JOB).`
- ☒ `P=(arrive -> work -> leave -> P).`
- ☐ `JOB=(arrive -> work -> leave).`
- ☐ *Aucune de ces réponses n'est correcte.*

**Question 11** ♣ Quel est le processus FSP correspondant à la description LTS suivante :




- ☐ `MEETING=(hello -> converse -> goodbye).`
- ☐ `P=(hello -> converse -> goodbye).`
- ☒ `MEETING=(hello -> converse -> goodbye -> STOP).`
- ☐ `MEETING=(hello -> converse -> goodbye -> MEETING).`
- ☐ `MEETING=(hello | converse | goodbye).`
- ☒ `P=(hello -> converse -> goodbye -> STOP).`
- ☐ *Aucune de ces réponses n'est correcte.*

**Question 12** ♣ Quel est le processus FSP correspondant à la description LTS suivante (DOUBLE) :




- ☒ `DOUBLE=(in [i:1..3] -> DOUBLE[i]), DOUBLE[j:1..3]=(out [2*j] -> DOUBLE).`
- ☒ `DOUBLE=(in [i:1..3] -> out [2*i] -> DOUBLE).`
- ☒ `DOUBLE(I=3)=(in [i:1..I] -> out [2*i] -> DOUBLE).`
- ☐ *Aucune de ces réponses n'est correcte.*

+4/5/38+

**Question 13** Donnez la définition de la propriété de sureté. ☐0 ☐1 ☐2 ☐4

La sureté est le fait que tout processus puissent entrer à un moment en zone critique et le programme ne contient ni pwit ni deadlock.

puisse entrer en section critique --> propriété de vivacité

-- pas de deadlock : OK

-- respect de l'invariant (ex: pas d'écrivain si déjà un écrivain ou un lecteur, pas de lecteur si un écrivain / une seul processus en section critique / etc.)