TD – Concurrence et exécution parallèle

Dans ce TD, nous allons nous familiariser avec les **threads** et la **concurrence.** Vous pouvez retrouver ces notions dans les chapitres 4.1 et 4.2 du livre.

Ce TD doit vous permettre de mieux comprendre les notions de threads et de concurrence en vue du TP noté.

Question 1 : Vous trouverez ci dessous un programme Oz qui implémente de l'exécution parallèle.

```
declare X0 X1 X2 X3
thread X1=1+X0 end
thread X3=X1+X2 end
{Browse [X0 X1 X2 X3]}
```

Expliquer ce qu'il se passe !
Détailler les affectations nécessaires pour débloquer ce programme !

Question 2 : Considérez les programmes ci-dessous et indiquez les valeurs affectées aux variables X, Y et Z en fin d'exécution. Vérifiez, seulement après, avec la commande Browse.

```
local X Y Z in
      thread if X==1 then Y=2 else Z=2 end end
      thread if Y==1 then X=1 else Z=2 end end
end

local X Y Z in
      thread if X==1 then Y=2 else Z=2 end end
      thread if Y==1 then X=1 else Z=2 end end
      X=2
end
```

Question 3 : Considérez le programme suivant :

Que vaut D ? Dans quel ordre les threads sont-ils créés ? Dans quels ordre sont-ils exécutés ?

```
declare A B C D in
thread D=C+1 end
thread C=B+1 end
thread A=1 end
thread B=A+1 end
{Browse D}
```

Question 4 : Créer un programme qui récupère les nombres d'une liste et qui calcul pour chacun de ces nombres le carré et la suite de Fibonacci.
Attention, l'ordinateur étant fatigué il ne pourra effectuer plus d'un calcul par seconde, utilisez la bonne instruction pour éviter de le surmener.
Expliquer comment se déroule l'exécution, ce qu'il se passe !

Question 5 : C'est bien gentil d'avoir pris soin de l'ordinateur mais maintenant je suis en retard ! Modifier le programme pour que l'ordinateur aille plus vite dans son exécution tout en respectant les délais implantés de 1000ms pour chaque opération.