Exercices sur le raisonnement par récurrence

Exercice 1G.1 :

On considère les suites (u_n) et (v_n) définies par :

$u_0 = 1$, $v_0 = 0$

et pour tout entier naturel n :

$u_{n+1} = 3u_n + 4v_n$ et $v_{n+1} = 2u_n + 3v_n$.

On cherche le premier rang à partir duquel u_n et v_n soient tous les deux supérieurs à 1000.

Écrire un algorithme qui affiche le premier couple $(u_n ; v_n)$ qui vérifie cette condition, en utilisant une boucle **Tant Que**.

Exercice 1G.2 :

1. Soit la suite (u_n) définie par $u_0 = 3$ et pour tout entier naturel n par $u_{n+1} = 2u_n + 5$.
   A l'aide d'un tableur, on obtient les valeurs des premiers termes de la suite (u_n).
   Quelle formule, étirée vers le bas, peut-on écrire dans la cellule A3 pour obtenir les termes successifs de la suite (u_n) ?
2. Soit la suite (v_n) définie par $v_0 = 3$ et pour tout entier naturel n par $v_{n+1} = 2nv_n + 5$.
   A l'aide d'un tableur, déterminer les premiers termes de la suite (v_n).

	A
1	u_n
2	3
3	11
4	27
5	59

Exercice 1G.3 : piège classique en programmation

On considère la suite (u_n) définie par $u_0 = 1$ et pour tout entier naturel n, $u_{n+1} = \left(\dfrac{n+1}{2n+4}\right) u_n$.

On admet que la limite de la suite (u_n) vaut 0.

Compléter l'algorithme ci-dessous, afin qu'il affiche la plus petite valeur de n pour laquelle $u_n \leq 10^{-5}$.

```
n ← 0
U ← 1
Tant que …
    n ← …
    U ← …
Fin Tant que
Afficher n
```

Réaliser alors un programme python déterminant la valeur du rang cherché.

CORRIGE – Notre Dame de La Merci – Montpellier – M. Quet

Exercice 1G.1 :

On considère les suites (u_n) et (v_n) définies par : $u_0 = 1$, $v_0 = 0$, et pour tout entier naturel n :

$$u_{n+1} = 3u_n + 4v_n \quad et \quad v_{n+1} = 2u_n + 3v_n .$$

On cherche le premier rang à partir duquel u_n et v_n soient tous les deux supérieurs à 1000.

*Écrire un algorithme puis un programme python qui affiche le premier couple $(u_n ; v_n)$ qui vérifie cette condition, en utilisant une boucle **Tant Que**.*

```
U = 1
V = 0
rang = 0
while U<= 1000 or V <= 1000:
   U , V = 3*U + 4*V , 2*U + 3*V
   rang += 1
print("Au rang " , rang , ", on obtient le couple (" , U , ";" , V , ")")
```

→ Au rang 5 , on obtient le couple (3363 ; 2378)

Exercice 1G.2 :

1. *Soit la suite (u_n) définie par $u_0 = 3$ et pour tout entier naturel n par $u_{n+1} = 2u_n + 5$.*

   A l'aide d'un tableur, on obtient les valeurs des premiers termes de la suite (u_n).

   Quelle formule, étirée vers le bas, peut-on écrire dans la cellule A3 pour obtenir les termes successifs de la suite (u_n) ?

   $= 2*A2+5$

	A
1	u_n
2	3
3	11
4	27
5	59

2. *Soit la suite (v_n) définie par $v_0 = 3$ et pour tout entier naturel n par $v_{n+1} = 2nv_n + 5$.*

   A l'aide d'un tableur, déterminer les premiers termes de la suite (v_n).

   Dans la case A3 : $= A2+1$
   Dans la case B3 : $= 2*A2*B2+5$

	A	B
1	*n*	v_n
2	0	3
3	1	15
4	2	65
5	3	395

Exercice 1G.3 : piège classique en programmation

On considère la suite (u_n) définie par $u_0 = 1$ et pour tout entier naturel n, $u_{n+1} = \left(\frac{n+1}{2n+4}\right)u_n$.

On admet que la limite de la suite (u_n) vaut 0.

Compléter l'algorithme ci-dessous, afin qu'il affiche la plus petite valeur de n pour laquelle $u_n \leq 10^{-5}$.
Réaliser alors un programme python déterminant la valeur du rang cherché.

L'erreur classique est liée à la place de la ligne 4 incrémentant le rang *n* : le programme aurait été plus simple si cette ligne avait été placée après le calcul de U.
On le visualise mieux sur le tableur suivant :

	A	B
1	n	u_n
2	0	1
3	1	$\frac{1}{4}$
4	2	$\frac{1}{12}$
5	3	$\frac{1}{32}$

Dans la case A3 : $=(A2+1)/(2*A2+4)*B2$

Dans l'algorithme, il ne faut donc pas injecter $=(n+1)/(2n+4)*U$ à la ligne 5, mais :

$=((n-1)+1)/(2(n-1)+4)*U$

Soit : $=n/(2n+2)*U$.

On obtient l'algorithme suivant :

$n \leftarrow 0$
$U \leftarrow 1$
Tant que U > 0,00001
  $n \leftarrow n+1$
  $U \leftarrow =n/(2n+2)*U$
Fin Tant que
Afficher n

On obtient le programme suivant :

```
U = 1
n = 0
while U>0.00001 :
   n += 1
   U = n/(2*n+2)*U
print("Au rang" , n , "la suite u devient inférieure à 0,00001" )
```

→Au rang 13 la suite u devient inférieure à 0,00001

Le programme suivant est plus simple :

```
U = 1
n = 0
while U>0.00001 :
   U = (n+1)/(2*n+4)*U
   n += 1
print("Au rang" , n , "la suite u devient inférieure à 0,00001" )
```