

Test 6

1. Écrivez une fonction récursive **nbChiffres(n)** qui retourne le nombre de chiffres contenus dans l'écriture décimale du nombre entier **n** (positif ou nul).

Par exemple :

- nbChiffres(7) retournera 1,
- nbChiffres(4725) retournera 4,
- nbChiffres(53) retournera 2.

Donnez-en une version non terminale et une version terminale.

2. Écrivez une fonction récursive **sommeChiffres(n)** qui retourne la somme des chiffres contenus dans l'écriture décimale du nombre entier **n** (positif ou nul).

Par exemple :

- sommeChiffres(0) retournera 0,
- sommeChiffres(4) retournera 4,
- sommeChiffres(51389) retournera 26.

Donnez-en une version non terminale et une version terminale.

3. Écrivez une fonction récursive **pairs(n)** qui retourne la liste des entiers (positifs ou nuls) pairs entre 0 et **n**.

Par exemple :

- pairs(7) retournera [0, 2, 4, 6],
- pairs(0) et pairs(1) retourneront [0],
- pairs(2) et pairs(3) retourneront [0, 2].

4. En adaptant la réponse à la question précédente, écrivez une fonction **multiples(a, n)** qui retourne la liste des multiples de **a** compris entre 0 et **n**.

Par exemple :

- multiples(2, 11) retournera [0, 2, 4, 6, 8, 10],
- multiples(7, 21) retournera [0, 7, 14, 21],
- multiples(5, 3) retournera [0].

5. Écrivez une fonction récursive **est_pair(n)** qui dit si oui ou non n est un entier pair.
6. Écrivez une fonction récursive **est_impair(n)** qui dit si oui ou non n est un entier impair.
7. Voici le code de deux fonctions **f** et **g**. Ces deux fonctions sont définies de façon récursive, mais « croisée », car $g(n)$ appelle $f(n-1)$ et de même $f(n)$ appelle $g(n-1)$.

```
1 def f(n) :  
2     if n == 0 :  
3         return True  
4     return g(n-1)  
5  
6 def g(n) :  
7     if n == 0 :  
8         return False  
9     return f(n-1)
```

- a) Représentez concrètement la pile d'appels pour les cas suivants : $f(4)$, $g(4)$, $f(5)$ et $g(5)$.
- b) En fait, comment pourrait-on renommer ces deux fonctions ?
8. Écrivez une fonction récursive **contient_voyelle(mot)** qui dit si oui ou non **mot** contient une voyelle. Vous pourrez vous servir, dans le code de la fonction, de la chaîne de caractères **voyelles = "aeiouyAEIOUY"**.
9. Écrivez une fonction récursive **est_en_minuscules(mot)** qui dit si oui ou non **mot** ne contient **que des lettres minuscules** (à l'exclusion de tout autre caractère, comme les espaces ou les ponctuations).

On considérera, comme cas d'arrêt, que le mot vide est écrit en minuscules.

Par exemple :

- `est_en_minuscules("")` retournera True,
- `est_en_minuscules("Bonjour")` retournera False,
- `est_en_minuscules("bonjour")` retournera True,
- `est_en_minuscules("hello world !")` retournera False.