Correction du DS d'Informatique n°3

Ce sujet est composé de deux exercices et un problème. Vous prendrez soin de bien justifier vos calculs.
Veillez à bien respecter l'indentation des programmes Python que vous écrirez sur votre copie.

Exercice 1.

1. Ecrire une fonction `recherche(L,x)` où `L` est une liste de nombres entiers et `x` est un entier et qui **renvoie** `True` si `x` est dans la liste `L` et qui renvoie `False` sinon. Par exemple, `recherche([1,2,3],2)` doit renvoyer `True`.

2. Écrire une fonction `moyenne(L)` où `L` est une liste non vide de nombres entiers qui renvoie la moyenne des éléments de la liste `L`.

3. Écrire une fonction `somme(L1,L2)` où `L1,L2` sont des listes supposées de même taille et composées de nombres entiers qui renvoie une liste contenant la somme terme à terme des listes `L1` et `L2`.
   Par exemple, `somme([1,2,3],[4,5,10])` doit renvoyer `[5,7,13]`.

4. Écrire une fonction `comptage(L)` où `L` est une liste de nombres entiers qui renvoie un dictionnaire dont les clés sont les valeurs des élements de `L` et dont les valeurs associées sont leurs nombres d'occurrences dans `L`.
   Exemple de comportement attendu :

```
>>>comptage([3,9,3,1,3,1,3])
{3:4; 9:1, 1:2}
>>>comptage([])
{}
```

5. Écrire une fonction `doublon(L)` où `L` est une liste de nombres entiers qui renvoie `True` si un élément apparait au moins deux fois dans la liste `L`, et `False` sinon.
   Indication : on pourra s'inspirer du fonctionnement de la fonction `comptage(L)` précédente.
   Exemple de comportement attendu :

```
>>>doublon([1,2,1,3])
True
>>>doublon([0,1,6])
False
>>>doublon([0,1,2,3,3,0,9,0])
True
```

Correction.

1.

```
def recherche(L,x):
    for e in L:
        if x == e:
            return True
    return False
```

2.

```
def moyenne(L):
    if len(L)==0:
        return 0
    s = 0
    for k in L:
        s+=k
    return s/len(L)
```

3.

```
def somme(L1,L2):
    S = []
    for i in range(len(L1)):
        S.append(L1[i]+L2[i])
    return S
```

4.

```
def comptage(L):
    d = {}
    for c in L:
        if c in d.keys():
            d[c] += 1
        else:
            d[c] = 1
    return d
```

5.

```python
def doublon(L):
    d = {}
    for c in L:
        if c in d.keys():
            return True
        else:
            d[c] = True
    return False
```

Exercice 2. *Récursivité*

1. On donne la fonction suivante :

```python
def s(m,n):
    """ m entier, n entier positif """
    if n==0:
        return m
    return s(m+1,n-1)
```

   Quel résultat renvoie cette fonction en fonction des entiers `m` et `n` ?

2. Écrire une fonction récursive `u(n)` qui renvoie le terme d'indice `n` ´ $\in \mathbb{N}$ de la suite $(u_n)_{n\in\mathbb{N}}$ définie par :

$$\begin{cases} u_0 = 0 \\ u_n = 1 + u_{n-1}^3 \quad \text{si } n \in \mathbb{N}^* \end{cases}$$

‘

Correction.

1. Après quelques essais, on conjecture que `s(m,n)` renvoie `m+n`. Montrons par récurrence sur $\mathbb{N}$ que, pour tout $n \in \mathbb{N}$, $\mathcal{P}(n)$ ="pour tout $m \in \mathbb{Z}$, $s(m,n) = m+n$" est vraie
   - Si $n = 0$, pour tout $m \in \mathbb{Z}$, $s(m,n) = m = m+n$.
   - Soit $n \in \mathbb{N}$. On suppose $\mathcal{P}(n)$ vraie. On a, pour tout $m \in \mathbb{Z}$:

$$s(m,n+1) = s(m+1,n) \underbrace{=}_{\text{H.R.}} m+1-n = m-(n+1).$$

   D'où le résultat.

2.

```python
def u(n):
    if n == 0:
        return 0
    else:
        return u(n-1)**3+1
```

Exercice 3. *CCP 2019*

Dans cet exercice "Algorithme de décomposition primaire d'un entier" (*Informatique pour tous*), on se propose d'écrire un algorithme pour décomposer un entier en produit de nombres premiers. Les algorithmes demandés doivent être écrits en langage **Python**. On sera très attentif à la rédaction et notamment à l'indentation du code.

On définit la valuation p-adique [de n] pour p nombre premier et n entier naturel non nul.
Si p divise n, on note $v_p(n)$ le plus grand entier k tel que p^k divise n.
Si p ne divise pas n, on pose $v_p(n) = 0$.
L'entier $v_p(n)$ s'appelle la valuation p-adique de n.

Q1 Écrire une fonction booléenne `estPremier(n)` qui prend en argument un entier naturel non nul n et qui renvoie le booléen `True` si n est premier et le booléen `False` sinon. On pourra utiliser le critère suivant : un entier $n \geq 2$ qui n'est divisible par aucun entier $d \geq 2$ tel que $d^2 \leq n$, est premier.

Q2 En déduire une fonction `liste_premiers(n)` qui prend en argument un entier naturel non nul n et renvoie la liste des nombres premiers inférieurs ou égaux à n.

Q3 Pour calculer la valuation 2-adique de 40, on peut utiliser la méthode suivante :

- 40 est divisible par 2 et le quotient vaut 20.
- 20 est divisible par 2 et le quotient vaut 10.
- 10 est divisible par 2 et le quotient vaut 5.
- 5 n'est pas divisible par 2.

La valuation 2-adique de 40 vaut donc 3.

Écrire une fonction `valuation_p_adique(n,p)` **non récursive** qui implémente cet algorithme. Elle prend en arguments un entier naturel n non nul et un nombre premier p et renvoie la valuation p-adique de n. Par exemple, puisque $40 = 2^3 \times 5$, `valuation_p_adique(40,2)` renvoie 3, `valuation_p_adique(40,5)` renvoie 1 et `valuation_p_adique(40, 7)` renvoie 0.

Q4 Écrire une deuxième fonction cette fois-ci **récursive** `val_p_adique(n,p)` qui renvoie la valuation p-adique de n.

Q5 En déduire une fonction `decomposition_facteurs_premiers(n)` qui calcule la décomposition en facteurs premiers d'un entier $n \geq 2$.
Cette fonction doit renvoyer la liste des couples $(p, v_p(n))$ pour tous les nombres premiers p qui divisent n.
Par exemple, `decomposition_facteurs_premiers(40)` renvoie la liste `[[2, 3], [5, 1]]`.

Correction.

Q1

```
def estPremier(n):
    d=2
    while d**2<=n:
        if n%d==0:
            return False
        d=d+1
    return True
```

Q2

```
def listePremiers(n):
    P=[]
    for k in range(2,n+1):
        if estPremier(k):
            P.append(k)
    return P
```

ou, en utilisant la syntaxe Python pour les listes (mais l'algo est équivalent) :

```
def listePremiers(n):
    return [k for k in range(2,n+1) if estPremier(k)]
```

Q3

```
def valuation_p_adique(n,p):
    puiss = 0
    a=n
    while a%p==0:
        puiss=puiss+1
        a=a//p
    return puiss
```

Q4

```
def valuation_p_adique(n,p):
    if n%p!=0:
        return 0
    return valuation_p_adique(n//p,p)+1
```

Q5

```python
def decomposition_facteurs_premiers(n):
    D=[]
    for p in liste_premiers(n):
        vp=valuation_p_adique(n,p)
        if vp!=0:
            D.append([p,vp])
    return D
```