II - TP 2

Fonctions récursives

1 Suites définies par une relation de récurrence

1.1 Suites géométriques

On dit qu'une suite (u_n) est géométrique lorsqu'il existe un nombre réel a et un nombre réel q tels que $u_0 = a$ et que pour tout $n \geq 1$ on ait $u_n = q \times u_{n-1}$. On écrit souvent ces informations de la manière suivante :

$$u_n = \begin{cases} a & \text{si } n = 0 \\ q \times u_{n-1} & \text{sinon.} \end{cases}$$

Écrire le code d'une fonction récursive `géométrique` qui prend en argument un entier `n`, deux nombres flottants `a` et `q`, et qui renvoie u_n, quand (u_n) est une suite géométrique de raison q et de premier terme a.

Code python

```
def géométrique(n, a, q):
    """ int, float, float -> float
    Renvoie u_n quand (u_n) est une suite géométrique de premier terme a
    ↪ et de raison q. """
    pass
```

Code python

```
print(géométrique(2, 3, 5))
```

Résultat

```
75
```

1.2 Suite arithmétiques

On dit qu'une suite (u_n) est arithmétique lorsqu'il existe un nombre réel a et un nombre réel q tels que $u_0 = a$ et que pour tout $n \geq 1$ on ait $u_n = q + u_{n-1}$. On écrit souvent ces informations de la manière suivante :

$$u_n = \begin{cases} a & \text{si } n = 0 \\ q + u_{n-1} & \text{sinon.} \end{cases}$$

Écrire le code d'une fonction récursive `arithmétique` qui prend en argument un entier `n`, deux nombres flottants `a` et `q`, et qui renvoie u_n, quand (u_n) est une suite arithmétique de raison q et de premier terme a.

Code python

```
def arithmétique(n, a, q):
    """ int, float, float -> float """
    pass
```

Code python

```
print(arithmétique(2, 3, 5))
```

Résultat

```
13
```

1.3 Fonction factorielle

Pour tout entier positif ou nul on définit la fonction factorielle de n de la manière suivante :

- Si $n = 0$, alors la factorielle de n vaut 1 ;
- Sinon, la factorielle de n vaut le produit des n premiers entiers strictement positifs : $1 \times 2 \times 3 \times \ldots \times n$.

On note $n!$ le nombre factorielle de n
Écrire une fonction python récursive `factorielle` qui prend en argument un entier positif `n` et qui renvoie le nombre $n!$.

Code python

```
def factorielle(n):
    """ int -> int """
    pass
```

Code python

```
print(factorielle(5))
```

Résultat

```
120
```

1.4 Suite de Fibonnacci

Wikipédia :

La suite doit son nom à Leonardo Fibonacci qui, dans un problème récréatif posé dans l'ouvrage Liber abaci publié en 1202, décrit la croissance d'une population de lapins :
« Quelqu'un a déposé un couple de lapins dans un certain lieu, clos de toutes parts, pour savoir combien de couples seraient issus de cette paire en une année, car il est dans leur nature de générer un autre couple en un seul mois, et qu'ils enfantent dans le second mois après leur naissance. »

On note F_n le nombre de lapins présents au début du n-ième mois. Jusqu'à la fin du deuxième mois, les lapins n'ont pas commencé à se reproduire, il y a donc $F_1 = F_2 = 1$ couple. Lorsque $n \geq 3$, on admet que le nombre F_n de couples lors du n-ième mois est donné par la formule suivante :

$$F_n = F_{n-2} + F_{n-1}$$

Question 1.

1. Calculer les 10 premiers nombres de fibonacci.
2. Écrire une fonction récursive `fibonacci` qui prend en argument un entier positif `n` et qui calcule le nombre F_n.

Code python

```
def fibonacci(n):
    """ int -> int
    n >= 1
    Détermine le n-ième nombre de Fibonacci avec F_0 = 0, F_1 = 1.  """
    pass
```

Code python

```
for i in range(1, 10):
    print(fibonacci(i), end = ' ')
```

Résultat

```
1 1 2 3 5 8 13 21 34
```

2 Fonctions récursives

2.1 Encadre

Question 2. L'objectif de cette question est d'écrire une version récursive de la fonction `encadre` qui étant donné un nombre entier `k` détermine la valeur du plus grand entier n que l'on puisse écrire en base 2 avec `k` bits.

1. Pour quelles valeurs de k est-il facile de calculer `encadre(k)` ?
2. On suppose dans cette question qu'un nombre entier n s'écrit en base 2 avec k bits.
   a. Quel est le plus grand entier n que l'on puisse écrire avec $k-1$ bits ?
   Exprimer n en fonction de k.
   b. Quel est le plus grand entier n que l'on puisse écrire avec k bits ?
   Exprimer n en fonction de k.
   c. En déduire l'expression de `encadre(k)` en fonction de `encadre(k-1)`.
3. En déduire le code d'une fonction récursive `encadre` qui réponde au problème de l'énoncé.

Code python

```
def encadre(k):
    """ int -> int
    Détermine le plus grand entier que l'on puisse écrire sur k bits """
    pass
```

Code python

```
print(encadre(4))
```

Résultat

```
15
```

2.2 Nombres de bits

Question 3. L'objectif de cette question est d'écrire une version récursive de la fonction `nombre_bits` qui étant donné un entier `n` détermine le nombre `k` de bits présents dans l'écriture en base 2 du nombre `n`.

1. Pour quelle(s) valeur(s) de n est-il facile de calculer `nombre_bits(n)` ?
2. On suppose dans cette question qu'un nombre entier n s'écrit en base 2 avec k bits.
   a. Avec combien de bits s'écrit en base 2 le nombre `n//2` ?
   Exprimer la réponse en fonction de k.
   b. En déduire l'expression de `nombre_bits(n)` en fonction de `nombre_bits(n//2)`.
3. En déduire le code d'une fonction récursive `nombre_bits` qui réponde au problème de l'énoncé.

Code python

```
def nombre_bits(n):
    """ int -> int
    Renvoie le nombre de bits nécessaires pour écrire n en base 2 """
    pass
```

Code python

```
print(nombre_bits(15))
print(nombre_bits(16))
```

Résultat

```
4
5
```

2.3 Appartient

L'objectif de cet exercice est d'écrire une version récursive de la fonction `appartient` qui étant donné un tableau (éventuellement vide) `tab` d'entiers et entier `e`, détermine si l'élément `e` est présent dans le tableau `tab`. On commence pour cela par écrire une fonction `appartient_aux` qui étant donné un tableau (éventuellement vide) `tab` d'entiers, un entier `e` et un entier `i` détermine si l'élément `e` est présent parmi les `i` premiers éléments du tableau `tab`. Si n est la taille du tableau `tab`, on supposera que $0 \leq i \leq n$.

1. Soient `tab` un tableau et `e` un élément.

   Que doit renvoyer l'instruction `appartient(tab, e, 0)` ?

2. On suppose dans cette questions que $1 \leq i \leq n$.

   a. Quel est l'indice du i-ème élément du tableau `tab` ?
   Exprimer votre réponse en fonction de i.

   b. Avec quels arguments faut-il appeler la fonction `appartient_aux` pour déterminer si l'élément `e` est présent parmi les $i-1$ premiers éléments du tableau ?

3. Décommenter et compléter le code ci-dessous.

Code python

```
def appartient_aux(tab, e, i):
    """ [int], int, int -> bool
    0 <= i <= len(tab)
    Détermine si l'élément e est présent parmi les i premiers éléments de
    ↪ tab. """
    # if i == 0:
        # return ..............................
    # else:
        # On teste si le i-ème élément du tableau est l'élément recherché.
        # if tab[............] == ............:
        #     return ..........................
        # Sinon on recherche l'élément e parmi les i-1 premiers éléments.
        # else:
        #     return ..........................

def appartient(tab, e):
    """ [int], int -> bool
    Détermine si l'élément e est présent dans le tableau tab. """
    # return ....................................
```

Code python

```
print(appartient([0, 1, 2, 3], 1))
print(appartient([0, 1, 2, 3], 4))
```

Résultat

```
True
False
```

2.4 Nombre d'occurrences

En vous inspirant de l'algorithme de l'exercice précédent, écrire une fonction `nombre_occurrences_aux` qui étant donné un tableau (éventuellement vide) `tab` d'entiers, un entier `e`, et un entier `i` détermine le nombre de fois où l'élément `e` est apparait parmi les `i` premiers éléments du tableau `tab`. Si n est la taille du tableau `tab`, on supposera que $0 \leq i \leq n$.

Code python

```
def nombre_occurrences_aux(tab, e, i):
    """ [int], int, int -> int
    Renvoie le nombre d'occurences de e parmi les i premiers éléments """
    pass
```