4TPM205U: Algorithmique des tableaux: feuille 8

Fonctions récursives

Travaux dirigés

Exercice 1 – Factorielle récursive

On montre que $n!$ est défini par :

$$n! = \begin{cases} 1 & \text{si } n = 0 \\ n * (n - 1)! & \text{si } n \geq 1 \end{cases}$$

Écrire une fonction `factorielleRecursive` utilisant cette définition.

Exercice 2

Les fonctions `f1` et `f2` sont définies comme suit :

<pre>def f1(n): if n == 0: return 1 return f1(n) + f1(n)</pre>		<pre>def f2(n): return f2(n-1) + f2(n-1)</pre>
--	--	--

Quel est le comportement de `f1(5)` ? de `f2(5)` ?

Exercice 3

Écrire une fonction récursive `sommeEltRecu(t,n)` qui calcule et renvoie la somme des éléments d'un tableau `t` contenant `n` nombres, $n > 0$.

Exercice 4

Une suite de nombres est *palindrome* si on peut la lire indifféremment de gauche à droite ou de droite à gauche. Soit `t` un tableau de `n` nombres, on veut tester s'il constitue une suite palindrome.

1. Écrire une fonction récursive `estPalindromeSectionRec(t,i,j)` qui renvoie `True` si la section du tableau `t` entre les indices `i` et `j` inclus constitue une suite palindrome et renvoie `False` sinon. On supposera, par convention, que la fonction renvoie `True` si $i > j$.
Exemples : Si `t=[3,7,2,1,1,2,5]` l'appel `estPalindromeSectionRec(t,2,5)` doit renvoyer `True`, alors que l'appel `estPalindromeSectionRec(t,0,6)` doit renvoyer `False`.
2. Écrire une fonction `estPalindromeTab(t,n)` qui fait appel à la fonction précédente pour tester si les `n` nombres contenus dans `t` constituent une suite palindrome.

Exercice 5

1. On considère les fonctions `test1` et `test2` suivantes :

```
def test1(n):
    if n == 0:
        print(0)
    else:
        print(n)
        test1(n - 1)

def test2(n):
    if n == 0:
        print(0)
    else:
        test2(n - 1)
        print(n)
```

Que font les fonctions `test1` et `test2`?

2. Soit maintenant les fonctions `test3` et `test4` qui s'appellent mutuellement :

```
def test3(n):
    if n == 0:
        print(0)
    else:
        print(n)
        test4(n - 1)

def test4(n):
    if n == 0:
        print(0)
    else:
        test3(n - 1)
        print(n)
```

Simuler leur exécution pour `n=3`.