Session d'exercices – Récursivité

Exercice 1: La suite de Fibonacci

[Difficulté: *]

Implémentez une fonction récursive `fibonacci(n)` pour calculer la suite de Fibonacci:

$U_n = U_{n-1} + U_{n-2}$, $U_1 = 1$, $U_0 = 0$ (chaque élément de la suite est la somme des deux précédents).

Testez-la. Par exemple, le résultat attendu pour $n = 5$ est 5, tant que pour $n = 8$ est 21.

Exercice 2: La fonction *max* élément d'une liste

[Difficulté: **]

Implémentez une fonction `recursive_max(n, tab)` de manière récursive. Cette fonction doit prendre en argument une liste d'entiers et la taille de cette liste, et retourner la valeur maximale de cette liste.

Testez-la en utilisant la liste suivante:

```
tab = [3, 6, 12, 1, 34, -3, 21, 8, 22, -13]
```

Le résultat attendu est 34.

Exercice 3 : La distance de Hamming

[Difficulté: **] Pour deux listes de même taille, *la distance de Hamming* est le nombre de positions où les valeurs diffèrent entre ces deux listes. Par exemple, la distance de Hamming entre les listes $[1, 2, 3, 4, 5]$ et $[1, 1, 3, 3, 5]$ est 2, puisqu'il y a deux positions où les valeurs ne sont pas les mêmes (les positions 1 et 3).

Écrivez la fonction `hamming_distance` qui trouve cette distance de manière récursive.

Testez-la en utilisant les listes suivants:

```
tab1 = [10,123,141,-123,3423,2,1,4,5,6,7,3,4,5,8,7,5,4,12,32]
tab2 = [1,12,141,-123,132,2,1,4,5,6,7,3,34,5,8,7,5,4,2,32]
```

Le résultat attendu est 5.

Hint : Réfléchissez bien à ce que doit être votre condition de terminaison (cas de base) avant de vous lancer dans le code.

Exercice 4: Factorielle

[Difficulté: *]

Implémentez une fonction `factorial(n)` de manière récursive.

Rappel: $n! = n \cdot (n-1)! = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 2 \cdot 1$

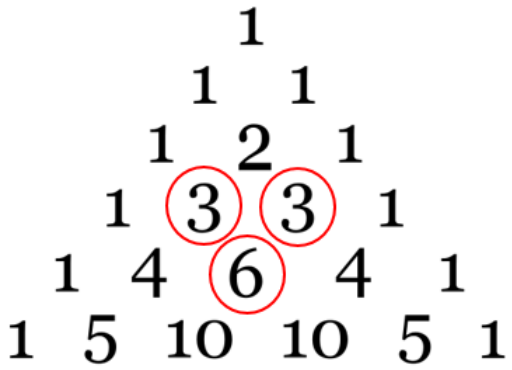
Testez la fonction avec la valeur de n dans l'intervalle $1 \leq n \leq 12$.

Exercice 5: Triangle de Pascal

[Difficulté: **]

Les rangées du triangle de Pascal sont conventionnellement numérotées en commençant par la rangée $row = 0$ en haut. Les éléments de chaque rangée sont numérotés de gauche à droite, en commençant par $col = 0$, et sont généralement décalés par rapport aux nombres des rangées adjacentes. Le premier et le dernier élément de chaque rangée est toujours 1. Les éléments intermédiaires sont égaux à la somme des deux éléments au-dessus d'eux.

Par exemple, l'élément à la rangée 4 et la colonne 2 est 6; 6 est égal à la somme de l'élément à la rangée 3 et colonne 1 et de l'élément à la même rangée et colonne 2 ($6 = 3 + 3$).



Le but de cet exercice est d'implémenter une fonction `pascals_triangle(row, col)` qui calcule l'élément du triangle de Pascal qui se trouve à la ligne *row* et à la colonne *col* ($row \geq col$) de manière **récursive**. Comparez le résultat avec l'expression suivante :

$$\texttt{pascals\_triangle(row, col)} = \frac{\texttt{factorial(row)}}{\texttt{factorial(col)} \cdot \texttt{factorial(row-col)}}$$