

► **Exercice 1.** Pour chacune des fonctions suivantes, l'exécuter à la main sur un exemple, dire pour quelles valeurs du paramètre n elle se termine, et quand c'est le cas ce qu'elle fait ou calcule.

```
public static int f(int n)
{
    if (n == 0) return 1;
    else return f(n+1);
}

public static int sommeBis(int n)
{
    if (n == 0) return 0;
    else {
        int result = sommeBis(n-1);
        result += n;
        return result;
    }
}

public static int g(int n)
{
    if (n <= 1) return 1;
    else return 1 + g(n-2);
}
```

► **Exercice 2.** Construire la forme récursive de la fonction $g()$ définie ci-dessous.

```
public static void g(int n){
    for(int i = 0; i < n; i++)
        System.out.println(i);
    return; }
```

► **Exercice 3.** Écrire une fonction récursive qui calcule la somme de nombres de 1 à n , si $n > 0$ et renvoie 0 sinon. Quelle est la complexité?

► **Exercice 4.** Donner un algorithme récursif pour calculer x^n , x et n positifs non nuls. Quelle est la complexité? Peut-on calculer x^n avec moins de multiplications?

► **Exercice 5.** Écrire une fonction récursive $\text{pgcd}(m, n)$ qui calcule le plus grand diviseur commun des deux entiers (non-négatifs) m et n .

► **Exercice 6. (Triangle de Pascal).**

Calculer directement par récursivité les combinaisons : $C_n^p = C_{n-1}^{p-1} + C_{n-1}^p$ pour n et p donnés, avec $0 < p < n$ Conditions limites : $C_0^0 = 1 = C_i^0 = C_i^i$. Quelle est la complexité ?

► **Exercice 7. (Chiffres en base 10).**

Écrire un algorithme récursif qui prends un paramètre n et qui teste si n contient au moins un zéro dans son écriture en base 10. On fait ici la convention que l'écriture en base 10 de zéro est zéro. Quelle est la complexité ?

► **Exercice 8. (Récursion mutuelle).**

On considère les fonctions mutuellement récursives suivantes :

```
def u(n):
    if n == 0:
        return 1
    else:
        return u(n-1)+v(n-1)
def v(n):
    if n == 0:
        return 0
    else:
        return 2*u(n-1)+v(n-1)
```

1. Calculer $u_0, u_1, u_2, u_3, v_0, v_1, v_2$ et v_3 .
2. Donner les récurrences vérifiées par les suites (u_n) et (v_n) et montrer que la suite (u_n) vérifie la récurrence, $u_{n+2} = 2u_{n+1} + u_n$.

► **Exercice 9.** Décrire une fonction récursive qui, étant donné un entier X , détermine la valeur la plus proche de X dans un tableau d'entiers.

► **Exercice 10.**

On considère la suite de Lucas définie par $L_0 = 2, l_1 = 1$, et pour $n > 0, L_{n+2} = L_{n+1} + L_n$.

1. Écrire une fonction récursive qui calcule L_n en fonction de n .

On admet que

$$L_n = \left(\frac{1 + \sqrt{5}}{2} \right)^n + \left(\frac{1 - \sqrt{5}}{2} \right)^n .$$

2. Quel est la complexité de cette algorithme ?
3. En utilisant une fonction à deux paramètres écrire un algorithme dont la complexité est $\Theta(n)$.
4. Peut-on faire mieux ?