I - Premiers exemples

Exercice 1: Fonction récursive à paramètre entier

1. Écrire une fonction récursive `puissanceRec(x,n)` renvoyant la puissance n d'un nombre réel x basée sur la formule naturelle : $\forall n \geqslant 1, x^n = x \times x^{n-1}$.
2. On souhaite améliorer la fonction précédente qui n'est pas très performante : le calcul de x^{20} nécessite par exemple 19 multiplications.
   On peut remarquer que :
   $$x^{20} = (x^{10})^2 = ((x^5)^2)^2 = ((x \times x^4)^2)^2 = ((x \times (x^2)^2)^2)^2.$$
   Ce calcul ne nécessite plus que 5 multiplications.
   Écrire une fonction `PuissanceRapide(x,n)` qui calcule x^n selon le principe suivant :
   $$x^0 = 1 \text{ et } \quad \forall n \geqslant 1, x^n = \begin{cases} x \times \left(x^{\frac{n-1}{2}}\right)^2 & \text{si} \quad n \text{ est impair} \\ \left(x^{\frac{n}{2}}\right)^2 & \text{si} \quad n \text{ est pair} \end{cases}$$

Exercice 2: Fonction récursive à paramètre chaîne de caractères

Écrire de façon récursive une fonction `miroir(chaine)` calculant le miroir d'une chaîne de caractères (mêmes lettres à l'envers).
Par exemple `miroir("miroir")` renvoie `"riorim"`

Exercice 3: Fonction récursive à paramètre un couple d'entiers

Si a, b sont deux entiers naturels, on a :
$$\text{pgcd}(a,b) = \begin{cases} a & \text{si} \quad b = 0 \\ \text{pgcd}(b,r) & \text{sinon} \end{cases}$$
avec r le reste de la division euclidienne de a par b.
Écrire une fonction récursive `pgcd(a,b)` renvoyant le pgcd des entiers a et b.

II - Utilisation de la dichotomie

Exercice 4: Recherche dichotomique

On considère une liste triée **dans l'ordre croissante** de n éléments.
Soit x entier ou un flottant.
Objectif :
Écrire une fonction renvoyant indiquant si la liste contient `x` ou non.
Si $n = 0$ ou $n = 1$, on obtient facilement le booléen souhaité.
Sinon, si $n > 1$, on teste l'élément du milieu de la liste `L[n//2]` : si cet élément est `x`, on renvoie `true` sinon si cette valeur est plus petite que `x` on cherche à droite de ce milieu, et dans le cas contraire à gauche.
Appliquer ce principe pour tester l'appartenance d'un entier ou d'un flottant dans une liste triée par ordre croissant.

Exercice 5: Décomposition en base 2

Écrire une fonction `decomposition` prenant en argument un entier positif N et renvoyant l'écriture de binaire de N sous la forme d'une chaîne de caractères.
On renverra la chaîne vide pour zéro.
On commencera par donner une relation simple entre `decomposition(n)` et `decomposition(n//2)`.

III - Prolongements

Exercice 6: Une suite et de l'algorithmique

On considère la suite $(u_n)_{n\geqslant 0}$ donnée par $u_0 = u_1 = 1$, $u_2 = 2$ et $\forall n \in \mathbb{N}$, $u_{n+3} = 2u_{n+2} - u_{n+1} + u_n$.

1. Écrire une fonction récursive de la variable entière N et qui renvoie la liste des $N+1$ premiers termes de la suite $(u_n)_{n\geqslant 0}$.
2. On appelle `mot` une liste composée de 0 et de 1. Écrire une fonction `mot` de la variable entière n et qui renvoie la liste de tous les mots de longueur n.
   Par exemple, `mot(2)` renvoie `[[0,0],[0,1],[1,0],[1,1]]`.
3. On appelle *bon mot* un mot tel que tous les 1 présents soient immédiatement suivis ou précédés d'un autre 1. Par exemple, `[0,1,1,0]` et `[1,1,0,1,1]` sont des bons mots alors que `[1,0,0,1,1]` et `[0,1,0]` n'en sont pas.
   Écrire une fonction `bonMot` qui prend comme argument un mot `m` et qui

renvoie un booléen indiquant si m est un bon mot ou pas.

4. Écrire une fonction de la variable entière N qui renvoie tous les bons mots de longueur N.
5. Vérifier que pour n compris entre 2 et 15, u_n est le nombre de bons mots de longueur n.

Exercice 7: Algorithme Glouton

On dispose de jetons ayant une valeur entière prise dans une liste de valeurs `[a,b,c,...]`. Le nombre de jetons pour chaque valeur de la liste est illimité. On cherche à payer une somme entière n avec ces jetons.
On veut obtenir exactement n.

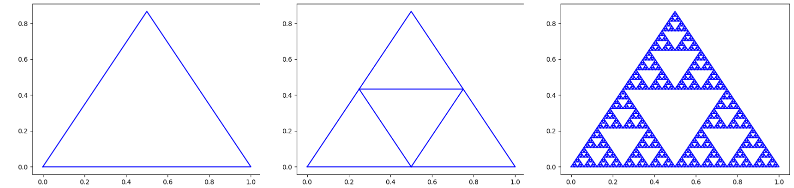
1. Définir une fonction `P5` d'argument n et qui donne le nombre de façons de payer la somme n avec des jetons de valeur 5. La fonction renvoie donc 1 si n est un multiple de 5 et 0 sinon.
2. Définir une fonction `P25` d'argument n qui donne le nombre de façons de payer la somme n avec des jetons dont les valeurs sont dans `[2,5]`. Afficher `P25(20)`.

   Indication : faire apparaître une boucle d'indice k et utiliser l'appel `P5(n-2k)`.
3. Définir une fonction `P125` d'argument n et qui donne le nombre de façons de payer la somme n avec des jetons dont les valeurs sont dans `[1,2,5]`. Afficher `P125(20)`.
4. Écrire une fonction `P` d'argument `V` et n où `V` est une liste de valeurs `V=[a,b,c,...]` et qui renvoie le nombre de façons de payer la somme n avec des jetons dont les valeurs sont dans `V`. Tester les appels `P([2,5],20)` ; `P([1,2,5],20)` ; `P([5,2,1],20)`.
5. Comparer `P([1,2,3,4,5,6],100)` et `P([6,5,4,3,2,1],100)`. Comment optimiser cette fonction ?

IV - Quelques figures fractales

Exercice 8

1. Écrire une fonction récursive permettant d'obtenir la figure fractale suivante :



2. Écrire une fonction récursive permettant d'obtenir la figure fractale suivante :