Exercice 1 *3 points*

1. Écrire une fonction récursive `puissance(x, n)` qui renvoie le nombre x^n.
2. Proposer une amélioration algorithmique de cette fonction, permettant une exécution plus rapide.

Exercice 2 *3 points*

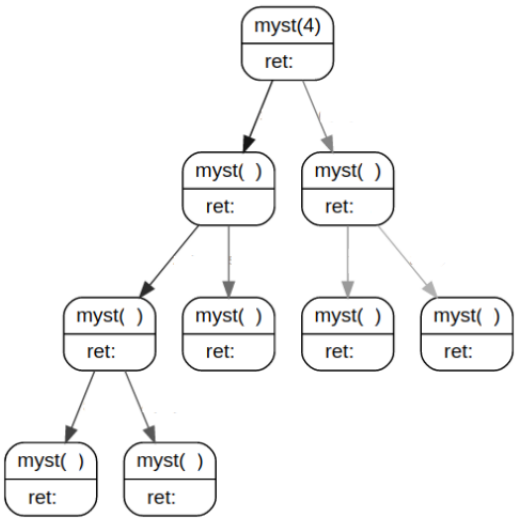
Écrire une fonction récursive `recherche(lst, m)` qui recherche la présence de la valeur `m` dans une liste triée (par ordre croissant) `lst`.
Cette fonction doit renvoyer un booléen.

Exercice 3 *3 points*

On considère la fonction `myst(n)` ci-dessous :

```
1  def myst(n):
2      if n == 0:
3          return 1
4      if n == 1:
5          return 2
6      else:
7          return myst(n - 1) * myst(n - 2)
```

1. Donner la relation entre `myst(n)`, `myst(n - 1)` et `myst(n - 2)`
2. Recopier et compléter l'arbre d'appel (voir ci-contre) lors du calcul de `myst(4)`.
   `ret` : signifie la valeur renvoyée par la fonction.



Exercice 4 *3 points*

On rappelle que le PGCD de deux nombres entiers a et b vérifie les propriétés suivantes :

- $pgcd(a, b) = pgcd(b, r)$, où r est le reste de la division euclidienne de a par b.
- $pgcd(b, 0) = b$

Écrire le code Python d'une fonction récursive `pgcd(a,b)` qui renvoie le PGCD de deux nombres entiers `a` et `b`.

Exercice 5 *4 points*

On rappelle que la suite de Fibonnaci est définie de la manière suivante :

- $F_0 = 0$ et $F_1 = 1$
- pour tout $n \in \mathbb{N}$, $F_{n+2} = F_{n+1} + F_n$

1. Écrire une fonction récursive `fibo(n)` qui renvoie la valeur de F_n pour tout entier naturel `n` passsé en paramètre.
2. Dessiner les différents états de la pile d'appel lors de l'exécution de l'instruction `fibo(3)`.

Exercice 6 *4 points*

La fonction récursive d'Ackermann (1896-1962, Allemagne), notée Ack dans la suite de l'exercice, est définie par :

$$\text{pour } m \in \mathbb{N}, n \in \mathbb{N}, \qquad Ack(m,n) = \begin{cases} n+1 & \text{si } m = 0 \\ Ack(m-1,1) & \text{si } m > 0 \text{ et } n = 0 \\ Ack(m-1, Ack(m, n-1)) & \text{si } m > 0 \text{ et } n > 0 \end{cases}$$

1. Calculer les valeurs suivantes.
   a. $Ack(0,1)$ **b.** $Ack(0,2)$ **c.** $Ack(1,0)$ **d.** $Ack(2,0)$ **e.** $Ack(1,2)$
2. Écrire le code Python de la fonction `ack(m,n)` qui renvoie le nombre $Ack(m,n)$.