Nom : 9 novembre 2020

Interro — Récursivité — Corrigé

Exercice 1 *(3 points)*

a. `mystere([10,9,11,42],3)` renvoie :

```
return 11 + mystere([10,9,11,42],2)
                |
        return 9 + mystere([10,9,11,42],1)
                        |
                return 10 + mystere([10,9,11,42],0)
                                |
                            return 0
```

Dessiner l'arbre des appels correspondants. On fait donc la somme de 11+9+10+0 = 30

b. La fonction mystere(t,i) fait la somme du contenu du tableau t de 0 jusqu'à i-1.

Exercice 2 *(3 points)*

La récursivité croisée permet d'appeler une fonction récursive depuis une autre fonction récursive. Par exemple, un nombre n est pair si $n-1$ est impair :

```python
def est_pair(n):
    if n == 0:
        return True
    return est_impair(n - 1)
```

```python
def est_impair(n):
    if n == 1:
        return True
    return est_pair(n - 1)
```

a. `est_pair(6)` : on refait un arbre d'appel.

```
return est_impair(5)
            |
      return est_pair(4)
              |
          return est_impair(3)
                  |
              return est_pair(2)
                      |
                  return est_impair(1)
                          |
                      return est_pair(0)
                              |
                          return True
```

Cette fonction renvoie donc True !

b. Le problème vient de la condition d'arrêt pour est_impair(n). En effet, est_impair(2) va appeler est_pair(1) qui va appeler est_impair(0).

Sauf que n=0 n'est pas pris en compte dans est_impair(0). D'où récursion infinie.

On peut réécrire la fonction est_impair :

```
def est_impair(n):
    if n == 0:
        return False
    return est_pair(n – 1)
```

Exercice 3 *(2 points)*

Il suffit de transcrire l'algorithme en Python :

```
def pgcd(a, b):
    if b == 0:
        return a
    return pgcd(b, a%b)
```

Il existe aussi un type d'expression dite expression ternaire assez utilisé en Javascript, et qui permet de "simplifier" la fonction :

```
def pgcd(a, b):
   return (a if b==0 else pgcd(b, a%b))
```

Exercice 4 (2 points)

```
def produitDesChiffres(n):
    if n<=9:
        return n
    else:
        return n%10 * produitDesChiffres(n//10)
```