Travaux dirigés #2 – Récursivité

Sauf mention contraire, les codes sont à donner en `Pseudo-code`.

Exercice 1

Yesterday is history, tomorrow is a mystery, today is a present.

1. Soit la fonction `mystery` suivante :

```
Fonction mystery(a, b) retourne entier
   Si b = 0 Alors
        retourner 0
   Si b % 2 = 0 Alors
        retourner mystery(a+a, b/2)
   retourner mystery(a+a, b/2) + a
```

Calculer `mystery(2, 25)` et `mystery(3, 11)`.

2. Étant donnés deux entiers positifs `a` et `b`, décrire ce que calcule `mystery(a, b)`.

3. Pour la fonction `mystery1` qui suit, décrire ce que `mystery1(a, b)` calcule pour deux entiers `a` et `b`.

```
Fonction mystery1(a, b) retourne entier
   Si b = 0 Alors
         retourner 1
   Si b % 2 = 0 Alors
         retourner mystery(a*a, b/2)
   retourner mystery(a*a, b/2) * a
```

4. Écrire les fonctions suivantes de façon récursive :

(a) `palindrome(mot)` qui retourne `Vrai` si `mot` est un palindrome et `Faux` sinon. Un palindrome est un mot qui est lu de manière identique de gauche à droite et de droite à gauche (*kayak* et *laval* sont des palindromes).

(b) `recherche(t,element) retourne entier` qui renvoie `Vrai` si `element` est présent dans le tableau et `Faux` sinon.

(c) `enum(n)` telle que pour `n` ≥ 0, la fonction `enum` retourne le tableau [0, 1, 2, …, n]. Si `n` < 0, la fonction retourne un tableau vide.

(d) `recherche_dichotomique(tableau,n)` qui prend en paramètre un tableau **trié** et un élément à rechercher dans le tableau et renvoie `Vrai` si l'élément est présent et `Faux` sinon. **La fonction doit exploiter le fait que le tableau est trié**.

5. Donner une version récursive terminale de toutes les questions précédentes.

6. Approfondissement. Donner une version récursive terminale calculant le dernier terme de la suite de Fibonacci.

Exercice 2

Les tours de Hanoï.

On dispose de 3 piquets désignés par `A, B, C` et de n disques de tailles différentes. Au départ, les n disques sont empilés du plus grand au plus petit sur le piquet numéro 1. Le but est de déplacer les n disques du piquet 1 vers le piquet 2 en respectant les règles suivantes :

- on ne déplace qu'un seul disque à la fois d'un piquet vers un autre ;
- un disque ne doit jamais être placé au dessus d'un disque plus petit que lui.

1. Combien de déplacements faut-il au total pour déplacer une pile de n disques ?
2. Écrire une fonction récursive `hanoi(A, B, C, n)` qui indique par des affichages les déplacements à effectuer pour transférer les n disques du piquet A vers le piquet B. Par exemple, pour $n = 3$, le programme indique :

```
A -> B
A -> C
B -> C
A -> B
C -> A
C -> B
A -> B
```

   qui signifie : prendre le disque au sommet du piquet `A` et le mettre sur le piquet `B`, puis prendre le disque au sommet du piquet `A` et le mettre sur le piquet `C` etc...
3. Écrire un programme en Python qui demande à l'utilisateur le nombre de disques à déplacer et qui fait appel à la fonction `hanoi` (traduite en Python) pour afficher les déplacements.

Exercice 3

Conjecture de {Syracuse, Collatz, Ulam, ... }.

La suite de Syracuse (...) est une suite définie de la manière suivante :

$$u_{n+1} = \begin{cases} \frac{u_n}{2} & \text{si } u_n \text{ est pair} \\ 3 * u_n + 1 & \text{sinon.} \end{cases}$$

Il est **conjecturé** que cette suite converge vers 1 : quelle que soit la valeur de départ, la valeur 1 finit par être atteinte. On note également que lorsque cette valeur est atteinte, la suite devient cyclique :

1 4 2 1 4 2 1 ...

On suppose donc que cette conjecture est vraie et que la valeur est 1 est toujours atteinte, quelle que soit la valeur de départ.

1. Donner le code d'une fonction récursive affichant toutes les valeurs de la suite de Collatz (...) depuis n'importe quelle valeur n.
2. Modifier ce code pour que la fonction retourne le tableau de toutes les valeurs successives.