DM d'informatique

Les exercices se valent progressifs :

— les deux premiers exercices sont considérés comme élémentaires, et doivent être réussis sans difficulté ;

— le troisième exercice se veut dans le prolongement d'exercices faits en TP, l'idée étant de coder autrement des programmes récursifs, en utilisant un dictionnaire pour stocker toutes les données intermédiaires et les restituer ;

— le dernier exercice, plus difficile, utilise pleinement l'utilisation d'un dictionnaire pour répondre à une énigme.

Exercice 1

1. Coder une fonction `absolue` qui prend en argument un nombre (de type `int` ou `float`) et rend sa valeur absolue du même type (c'est-à-dire que `absolue(-3)=3` (type `int`) et non `3.` (type `float`).
   Indication : on pourra utiliser un `if`.
2. Écrire une fonction `nombrediviseurs` qui prend en argument un entier naturel non nul `n` (de type `int`) et rend le nombre de ses diviseurs positifs. Transformer cette fonction en une fonction `sommediviseurs` qui prend le même argument en entrée et en rend la somme des diviseurs positifs.
   Indication : on pourra utiliser un `for` et des restes de divisions euclidiennes.
3. Coder deux fonctions `partieentiereiter` (itérative) et `partieentiererec` (récursive) qui prennent chacune en argument un nombre (de type `int` ou `float`) et rendent sa partie entière (de type `int`). On pourra commencer à traiter le cas où le nombre en entrée est positif ou nul. On prendra garde au fait que l'on doit avoir `partieentiereiter(-2.9)=-3` (et pareil pour la version récursive).
   Indication : on pourra utiliser un `if` dans les deux cas pour gérer le signe de l'entrée, et un `if` pour la version récursive ou un `while` pour la version récursive dans la suite du programme.

Exercice 2

1. Écrire une fonction `appartient` qui prend en argument une liste `L` et un objet `l` et rend `True` si `a` est un élément de `L` et `False` sinon. Modifier cette fonction pour qu'elle rende :
   — le premier indice où `a` apparaît dans `L` à la place de `True` ;
   — le nombre de fois que `a` apparaît dans `L` (et rendra donc `0` à la place de `False`) ;
   — la liste des indices où `a` est dans `L` (et rendra donc la liste vide `[]` à la place de `False`).
2. Écrire une fonction `somme` qui prend en argument une liste `L` de nombres et rend la somme de ses éléments. Modifier la fonction en `sommeindices` qui prend en plus en argument deux indices `j` et `k` et rend la somme des éléments de `L` dont les indices sont entre `j` et `k` (dans cet ordre, et au sens large).
3. Écrire une fonction `esttriee` qui prend en argument une liste `L` de nombres et rend `True` si les éléments de `L` sont rangés dans l'ordre croissant et `False` sinon. Écrire une fonction `croissancerapide` qui prend en argument une liste `L` de nombres et rend `True` si chaque élément de `L` est plus grand que la somme de ses éléments précédents, et `False` sinon.

Exercice 3

1. Écrire une fonction `fibodico` qui prend en argument un entier n calcule récursivement le n-ème terme de la suite de la suite, en stockant dans un dictionnaire tous les termes calculés pendant le processus. Le programme rendra le dictionnaire de toutes les valeurs calculées, dont les clés sont les entiers i pour lesquels F_i a été calculé, et les définitions sont les valeurs F_i correspondantes. Par exemple, pour $n = 4$, le programme rend le dictionnaire `{ 0: 0, 1: 1, 2: 1, 3: 2, 4: 3 }` .
2. Écrire une fonction `romaindico` qui prend en argument une chaîne de caractère codant pour un chiffre romain inférieur à 5000 et calcule sa valeur de manière récursive. Le programme rendra le dictionnaire de toutes les valeurs calculées, dont les clés sont les chaînes de caractères dont on a déterminé la valeur lors de la descente récursive, et les définitions sont les valeurs correspondantes. Par exemple, avec en entrée `'XCIX'`, le programme rend le dictionnaire `{'X': 10, 'IX': 9, 'CIX': 109, 'XCIX': 99 }` . Adapter ce programme pour qu'il rende le dictionnaire de tous les nombres jusqu'à 4999, des deux manières suivantes :
   — en déterminant toutes les écritures possibles de nombres entre 1 et 4999 (on vérifiera bien qu'il y a le bon nombre d'écritures différentes) ;
   — en codant de manière inverse les écritures en chiffres romain de tous les entiers entre 1 et 4999, et en appelant la fonction précédente sur ces écritures.

Exercice 4

Alceste, Bérénice et Célestin discutent :

A. - J'ai choisi deux entiers n, m tels que $1 < n < m$ et $n + m \leqslant 100$. Bérénice, je vais te dire leur produit. Et Célestin je vais te dire leur somme. Et vous devez essayer de deviner les nombres.
Alceste dit donc, comme annoncé, le produit $p = n \times m$ à Bérénice et la somme $s = n + m$ à Célestin.

B. - Je ne peux pas déduire les nombres.

C. - Je le savais !

B. - Dans ce cas je connais les deux nombres.

C. - Et moi aussi du coup !

Que valent n et m ?
Indication : on pourra utiliser un dictionnaire pour éliminer les couples (n, m) impossibles.