DS 2

Informatique de tronc commun, première année

Julien Reichert

Pour tous les exercices, le langage Python est imposé.

Exercice 1 : Écrire une fonction prenant en entrée une liste de listes `L` contenant des nombres et qui retourne la somme de tous les nombres dans toutes les listes de `L`.

Par exemple, la fonction retourne `26` pour la liste de listes `[[2, 5, 4], [3, 6], [4], [2]]`.

Exercice 2 : Écrire une fonction prenant en entrée une liste `l` de couples et un indice `i` entre `0` (inclus) et la taille de `l` (exclue) et qui retourne une liste formant une permutation de la liste `l`, selon la règle suivante : d'abord on met le couple à l'indice `i`, puis les couples dans la liste dont le premier élément est égal à celui à l'indice `i` dans l'ordre dans lequel ils figurent dans `l`, puis tous les autres couples de `l` en préservant également leur ordre.

Par exemple, la fonction retourne `[(3, 4), (3, 5), (3, 0), (2, 3), (1, 0), (2, 1), (2, 5)]` pour la liste `[(2, 3), (1, 0), (2, 1), (3, 5), (3, 4), (3, 0), (2, 5)]` et l'indice `4`.

Exercice 3 : Écrire une fonction prenant en entrée un entier naturel `n` et imprimant la figure de la forme ci-dessous sur `2*n-1` lignes (ici pour `n` valant `6`.

```
******
*****
****
***
**
*
**
***
****
*****
******
```

Exercice 4 : Écrire une fonction prenant en argument une liste supposée croissante et une valeur quelconque et déterminant si cette valeur est dans la liste. La fonction pourra être récursive ou non, mais toute technique ne procédant pas à une dichotomie sera refusée.

Exercice 5 : Écrire une fonction prenant en argument une liste supposée croissante et une valeur quelconque et déterminant combien de fois cette valeur est dans la liste. Les conditions sont les mêmes que pour l'exercice précédent. Noter que réussir pleinement cet exercice et utiliser la fonction ici obtenue pour l'exercice précédent donnera tous les points.

Exercice 6 : Écrire une fonction tentant de résoudre par un algorithme glouton le problème de chercher la plus grande somme que l'on peut obtenir en suivant un chemin dans une liste de listes, toutes de même taille, depuis le premier indice de la première liste et jusqu'au dernier indice de la dernière liste, en avançant à chaque étape d'un indice dans la liste actuelle ou au même indice dans la liste suivante.

Par exemple pour la liste

$$\begin{matrix}[[0, & 10, & 3, & 5], \\ [3, & 1, & 9, & 2], \\ [2, & 3, & 0, & 0]]\end{matrix}$$

la somme calculée sera 24, en allant successivement à droite (avancer d'un indice), à droite, en bas (en allant à la liste suivante), à droite et finalement en bas (plus le choix).

Exercice 7 : Cet algorithme résout-il le problème en trouvant la solution optimale ? Si non, donner un contre-exemple.

Exercice 8 : Expliquer ce que la fonction ci-dessous fait, en donnant par exemple le détail de son exécution sur une petite liste.

```
def fonction(l):
    rep = []
    for i in range(len(l)):
        nombre = l[i]
        ilyamieux = False
        for j in range(i):
            nombrebis = l[j]
            if nombrebis >= nombre:
                ilyamieux = True
        if not ilyamieux:
            rep.append(i)
    return rep
```

Pour information ou rappel, un tel exercice ne demande pas de paraphraser ligne par ligne mais bien d'interpréter le travail de la fonction dans sa globalité.