DS 3

Informatique de tronc commun, première année

Julien Reichert

Exercice 1 : Écrire une fonction prenant en entrée une liste de listes `L` et qui retourne le plus fréquent de tous les éléments dans toutes les listes de `L`. Le comportement est au choix en cas d'égalité.

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

Exercice 2 : Écrire une fonction prenant en entrée deux listes croissantes (propriété qu'on ne vérifiera pas) et qui retourne la liste croissante rassemblant les deux listes en argument.

Par exemple, la fonction retourne `[1, 2, 3, 4, 5, 6, 7, 9]` pour les listes `[2, 3, 5]` et `[1, 4, 6, 7, 9]`.

Exercice 3 : Écrire une fonction prenant en entrée une liste de couples (chaîne de caractères, valeur) et qui établit un classement général en tenant compte de possibles égalités. Il s'agit de renvoyer une liste de couples contenant en deuxième composante les chaînes de caractères dans l'ordre décroissant des valeurs associées, et en première composante le classement. L'algorithme de tri sera au choix mais à écrire soi-même.

Par exemple, la fonction retourne `[(1, "b"), (2, "a"), (3, "d"), (4, "c"), (4, "e")]` pour la liste `[("a", 42), ("b", 57), ("c", 3), ("d", 20), ("e", 3)])`.

Exercice 4 : Adapter la fonction précédente pour que la première composante ne soit plus simplement le classement mais la plage de classement occupée en tant que couple en cas d'égalité.

Par exemple, la fonction retourne `[(1, "b"), (2, "a"), (3, "d"), ((4, 5), "c"), ((4, 5), "e")]` pour la liste `[("a", 42), ("b", 57), ("c", 3), ("d", 20), ("e", 3)])`.

On peut se contenter de mentionner les lignes à changer par rapport à la fonction précédente.

Exercice 5 : Expliquer ce que la fonction ci-dessous fait, en donnant le détail de son exécution sur une petite liste.

```
def fonction(l):
    rep = []
    seuil = max(l) + 1
    while len(rep) < len(l):
        maxi = None
        nombre = 0
        for x in l:
            if x < seuil and (maxi == None or x >= maxi):
                if x == maxi:
                    nombre += 1
                else:
                    nombre = 1
                    maxi = x
        rep += [maxi] * nombre
        seuil = maxi
    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.

Exercice 6 : On se propose d'écrire une fonction qui détermine s'il existe une tranche d'une liste **d'éléments tous positifs ou nuls** dont la somme vaut une certaine valeur et qui la localise.

Le code ci-dessous est fourni :

```
def ssc(l, s):
    assert s >= 0
    n = len(l)
    debut = 0
    fin = 0
    ts = 0
    while True:
        if ts == s:
            return (debut, fin)
        if ts > s:
            ts -= l[debut]
            debut += 1
        elif fin < n:
            ts += l[fin]
            fin += 1
        else:
            return -1
```

Question 6.1 : Quelle est à tout moment la valeur de `ts` en fonction d'autres paramètres intervenant dans la fonction ?

Question 6.2 : Prouver (en déduire...) qu'à tout moment `debut` est inférieur ou égal à `fin`.

Question 6.3 : En déduire un variant qui permet de garantir la terminaison de la fonction.

Question 6.4 : Prouver à partir des réflexions précédentes la correction de la fonction.

Question 6.5 : Écrire une fonction bien plus simple qui résout le problème. Quel est cependant l'avantage de la version proposée ?