BACCALAUREAT

SESSION 2022

Épreuve de l'enseignement de spécialité

NUMERIQUE et SCIENCES INFORMATIQUES

Partie pratique

Classe Terminale de la voie générale

Sujet n°30

DUREE DE L'EPREUVE : 1 heure

Le sujet comporte 3 pages numérotées de 1 / 3 à 3 / 3
Dès que le sujet vous est remis, assurez-vous qu'il est complet.

Le candidat doit traiter les 2 exercices.

EXERCICE 1 (4 points)

Programmer la fonction `fusion` prenant en paramètres deux tableaux non vides `tab1` et `tab2` (type `list`) d'entiers, chacun dans l'ordre croissant, et renvoyant un tableau trié dans l'ordre croissant et contenant l'ensemble des valeurs de `tab1` et `tab2`.

Exemples :

```
>>> fusion([3, 5], [2, 5])
[2, 3, 5, 5]
>>> fusion([-2, 4], [-3, 5, 10])
[-3, -2, 4, 5, 10]
>>> fusion([4], [2, 6])
[2, 4, 6]
```

EXERCICE 2 (4 points)

Les chiffres romains sont un système ancien d'écriture des nombres.

Les chiffres romains sont: I, V, X, L, C, D, et M.
Ces symboles représentent respectivement 1, 5, 10, 50, 100, 500, et 1000 en base dix.

Lorsque deux caractères successifs sont tels que le caractère placé à gauche possède une valeur supérieure ou égale à celui de droite, le nombre s'obtient en additionnant le caractère de gauche à la valeur de la chaîne située à droite.

Ainsi, `"XVI"` est le nombre 16 car X + VI = 10 + 6.

Lorsque deux caractères successifs sont tels que le caractère placé à gauche possède une valeur strictement inférieure à celui de droite, le nombre s'obtient en retranchant le caractère de gauche à la valeur de la chaîne située à droite.

Ainsi, `"CDIII"` est le nombre 403 car DIII – C = 503 – 100.

On dispose d'un dictionnaire `dico`, à compléter, où les clés sont les caractères apparaissant dans l'écriture en chiffres romains et où les valeurs sont les nombres entiers associés en écriture décimale.

On souhaite créer une fonction récursive `rom_to_dec` qui prend en paramètre une chaîne de caractères (non vide) représentant un nombre écrit en chiffres romains et renvoyant le nombre associé en écriture décimale :

```
def rom_to_dec (nombre):
      """ Renvoie l'écriture décimale du nombre donné en chiffres romains """

      dico = {"I":1, "V":5, ...}
      if len(nombre) == 1:
            return ...

      else:

    ### on supprime le premier caractère de la chaîne contenue dans la
variable nombre et cette nouvelle chaîne est enregistrée dans la variable
nombre_droite
            nombre_droite = nombre[1:]

            if dico[nombre[0]] >= dico[nombre[1]]:
                  return dico[nombre[0]] + ...
            else:
                  return ...


assert rom_to_dec("CXLII") == 142
```

```
"""
Author: Pascal Padilla
Source: correction de l'exercice 1 du sujet 30 des épreuves pratiques NSI 2022

Remarques:
    * algorithme classique au c¿ur du tri fusion ! (merge sort)
"""


def fusion(tab1: list, tab2: list) -> list:
    """
    Tests et Exemples:
    >>> fusion([3, 5], [2, 5])
    [2, 3, 5, 5]
    >>> fusion([-2, 4], [-3, 5, 10])
    [-3, -2, 4, 5, 10]
    >>> fusion([4], [2, 6])
    [2, 4, 6]
    """
    # nombre d'éléments de chaque tableaux
    n_1 = len(tab1)
    n_2 = len(tab2)

    # création d'un tableau de taille égale à n1 + n2
    # intialement remplis avec des `None`
    tab_fusion = [None] * (n_1 + n_2)

    # BOUCLE: parcours de tab1 et tab2
    # -> invariant:
    #   * tab1[0 .. i-1] a été parcouru
    #   * tab2[0 .. j-1] a été parcouru
    #   * tab_fusion[0 .. i+j-1] est bien rempli
    i = 0
    j = 0

    # -> condition d'arrêt:
    #   * tab1 est complètement parcouru ET tab2 aussi
    #   * => donc i vaut n1 ET j vaut n2
    while not(i == n_1 and j == n_2):
        # cas où tab1 est complètement parcouru
        if i == n_1:
            # remplissage avec tab2
            tab_fusion[i+j] = tab2[j]
            j = j + 1

        # cas où tab2 est complètement parcouru
        elif j == n_2:
            # remplissage avec tab1
            tab_fusion[i+j] = tab1[i]
            i = i + 1

        # cas où tab1 et tab2 ne sont pas entièrement parcouru
        else:
            # l'élément courant de tab1 et plus petit que celui de tab2
            if tab1[i] < tab2[j]:
                tab_fusion[i+j] = tab1[i]
                i = i + 1

            # l'élément courant de tab2 est plus petit ou égal à celui de tab1
            else:
                tab_fusion[i+j] = tab2[j]
                j = j + 1

    # fin BOUCLE
    # tab1 et tab2 sont entièrement parcouru
    # et tab_fusion est correct
    return tab_fusion


# Vérification par doctest
from doctest import testmod
testmod()
```

```
"""
Author: Pascal Padilla
Source: correction de l'exercice 2 du sujet 30 des épreuves pratiques NSI 2022

Remarques:
    * l'argument `nombre` est trompeur : ce n'est pas un entier mais
    une chaîne de caractère (c'est un nombre...romain ;))
    * attention au sens de la soustraction pour le cas de la soustraction
"""


def rom_to_dec (nombre):

    """ Renvoie l¿écriture décimale du nombre donné en chiffres romains """

    # correspondance entre un caractère romain (la clé) et sa valeur
    dico = {"I":1, "V":5, "X": 10, "L": 50, "C": 100, "D": 500, "M": 1000}

    # cas de base : il n'y a pas d'addition ou de soustraction à faire
    if len(nombre) == 1:
        # renvoie de la valeur du caractère
        return dico[nombre]

    else:
        ### on supprime le premier caractère de la chaîne contenue dans la variable
nombre
        ### et cette nouvelle chaîne est enregistrée dans la variable nombre_droite
        nombre_droite = nombre[1:]

        # cas où il faut additionner (chiffre de gauche >= au chiffre de droite)
        if dico[nombre[0]] >= dico[nombre[1]]:
            # appel récursif: la valeur décimale est égale à la somme
            # de la valeur du chiffre de gauche et de la valeur de
            # l'ensemble des chiffres romains de droite
            return dico[nombre[0]] + rom_to_dec(nombre_droite)

        # cas de la soustraction: la valeur finale est égale à la
        # différence de la valeur de l'ensemble de droite
        # par la valeur du chiffre de gauche
        else:
            return rom_to_dec(nombre_droite) - dico[nombre[0]]



# Vérification par une assertion
assert rom_to_dec("CXLII") == 142

# Vérification par un affichage
print(rom_to_dec("CXLII"))
```