BACCALAUREAT

SESSION 2022

Épreuve de l'enseignement de spécialité

NUMERIQUE et SCIENCES INFORMATIQUES

Partie pratique

Classe Terminale de la voie générale

Sujet n°11

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)

Écrire une fonction `recherche` qui prend en paramètres un tableau `tab` de nombres entiers triés par ordre croissant et un nombre entier `n`, et qui effectue une recherche dichotomique du nombre entier `n` dans le tableau non vide `tab`.

Cette fonction doit renvoyer un indice correspondant au nombre cherché s'il est dans le tableau, `-1` sinon.

Exemples:

```
>>> recherche([2, 3, 4, 5, 6], 5)
3
>>> recherche([2, 3, 4, 6, 7], 5)
-1
```

EXERCICE 2 (4 points)

Le codage de César transforme un message en changeant chaque lettre en la décalant dans l'alphabet.

Par exemple, avec un décalage de 3, le A se transforme en D, le B en E, ..., le X en A, le Y en B et le Z en C. Les autres caractères ('!',' ?'…) ne sont pas codés.

La fonction `position_alphabet` ci-dessous prend en paramètre un caractère `lettre` et renvoie la position de `lettre` dans la chaîne de caractères `ALPHABET` s'il s'y trouve et `-1` sinon.

La fonction `cesar` prend en paramètre une chaîne de caractères `message` et un nombre entier `decalage` et renvoie le nouveau message codé avec le codage de César utilisant le décalage `decalage`.

```
ALPHABET='ABCDEFGHIJKLMNOPQRSTUVWXYZ'

def position_alphabet(lettre):
    return ALPHABET.find(lettre)

def cesar(message, decalage):
    resultat = ''
    for ... in message :
        if lettre in ALPHABET :
            indice = ( ... )%26
            resultat = resultat + ALPHABET[indice]
        else:
            resultat = ...
    return resultat
```

Compléter la fonction `cesar`.

Exemples :

```
>>> cesar('BONJOUR A TOUS. VIVE LA MATIERE NSI !',4)
'FSRNSYV E XSYW. ZMZI PE QEXMIVI RWM !'


>>> cesar('GTSOTZW F YTZX. ANAJ QF RFYNJWJ SXN !',-5)
'BONJOUR A TOUS. VIVE LA MATIERE NSI !'
```

```python
"""
Author: Pascal Padilla
Source: correction de l'exercice 1 du sujet 11 des épreuves pratiques NSI 2022
"""

from doctest import testmod


def recherche(tab: list, n: int) -> int:
    """Recherche dichotomique dans un tableau trié tab
    de l'élément n.

    Args:
        tab (list): tableau trié de nombres entiers non vide
        n (int): nombre entier à chercher

    Returns:
        int: si n est dans tab, renvoie l'indice
             sinon renvoie -1

    Exemples et tests:
    >>> recherche([2, 3, 4, 5, 6], 5)
    3
    >>> recherche([2, 3, 4, 6, 7], 5)
    -1
    """
    i = 0
    j = len(tab) - 1

    # Invariants à chaque tour de boucle :
    #   * les éléments avant l'indice i sont inférieurs à n
    #   * les éléments après l'indice j sont supérieurs à n
    #   * ¿ si n est dans tab, alors il est dans tab[i..j]
    # Initialisation de i et de j
    #   * i ¿ 0 et j ¿ dernier indice
    # Condition d'arrêt de la boucle:
    #   * soit on trouve n
    #   * soit tab[i..j] devient l'intervalle vide
    #       ¿ i dépasse j
    #       ¿ i > j

    while not i > j:
        # principe de la recherche dichotomique
        # comparer n avec la valeur médiane de tab[i..j]

        # indice médian
        i_median = (i + j) // 2

        # comparaison

        # si on trouve la valeur cherchée
        # on arrête le programme/la boucle
        # et on renvoie l'indice
        if tab[i_median] == n:
            return i_median

        # non trouvé donc
        #   * si n est supérieur alors on cherche dans la
        #       zone qui succède à l'élément médian
        if n > tab[i_median]:
            i = i_median + 1

        #   * sinon n est inférieur, alors on cherche dans la
        #       zone qui précède l'élément médian
        else:
            j = i_median - 1

    # Variant de boucle : l'intervalle [i..j] diminue à chaque fois

    # si on sort de la boucle,
    # c'est que l'élément n'a pas été trouvé
    return -1
```

```python
# Tests avec des assertions
assert recherche([2, 3, 4, 5, 6], 5) == 3
assert recherche([2, 3, 4, 6, 7], 5) == -1

# Tests avec la bibliothèque doctest
testmod()
```

```python
"""
Author: Pascal Padilla
Source: correction de l'exercice 2 du sujet 11 des épreuves pratiques NSI 2022
"""

from doctest import testmod


# définition de toutes les lettres acceptables
# dans une chaîne de caractère
# (qui se parcourt avec la même syntaxe que les tableaux)
#
# écrire une variable en majuscule pour
# signifier que c'est une constante
ALPHABET = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'


def position_alphabet(lettre):
    """Indice de `lettre` si elle est présente et -1 sinon

    Args:
        lettre (str): lettre à chercher

    Returns:
        int: indice de la lettre dans ALPHABET
                ou -1 si elle n'y est pas
    """
    # la méthode `find` s'applique sur les chaînes de caractères
    return ALPHABET.find(lettre)


def cesar(message, decalage):
    """
    Exemples et doctests
    >>> cesar('BONJOUR A TOUS. VIVE LA MATIERE NSI !',4)
    'FSRNSYV E XSYW. ZMZI PE QEXMIVI RWM !'
    >>> cesar('GTSOTZW F YTZX. ANAJ QF RFYNJWJ SXN !',-5)
    'BONJOUR A TOUS. VIVE LA MATIERE NSI !'
    """
    # initialisation du message codé
    resultat = ''

    # parcourt toutes les lettres de message, une par une
    # en commençant par la gauche.
    # la lettre courante est dans la variable `lettre`
    for lettre in message :

        # la lettre doit être codée
        if lettre in ALPHABET :

            # ajoute à l'indice de la lettre dans
            #   l'alphabet un décalage
            # l'appel à modulo 26 permet en cas de débordement
            #   de repartir à l'indice 0
            indice = ( position_alphabet(lettre) + decalage )%26

            # ajoute la lettre codée dans le codage du message
            resultat = resultat + ALPHABET[indice]

        # la lettre n'est pas dans l'alphabet et
        # ne doit donc pas être codée : on la laisse
        # donc inchangée (cas du `!` par exemple)
        else:
            resultat = resultat + lettre

    # renvoie du message codé
    return resultat


# Exemples
print(cesar('BONJOUR A TOUS. VIVE LA MATIERE NSI !',4))
print(cesar('GTSOTZW F YTZX. ANAJ QF RFYNJWJ SXN !',-5) )
```

```
# Tests avec des assertions
assert cesar('BONJOUR A TOUS. VIVE LA MATIERE NSI !',4) == 'FSRNSYV E XSYW. ZMZI PE
QEXMIVI RWM !'
assert cesar('GTSOTZW F YTZX. ANAJ QF RFYNJWJ SXN !',-5) == 'BONJOUR A TOUS. VIVE LA
 MATIERE NSI !'


# Tests avec la bibliothèque doctest
testmod()
```