BACCALAUREAT

SESSION 2022

Épreuve de l'enseignement de spécialité

NUMERIQUE et SCIENCES INFORMATIQUES

Partie pratique

Classe Terminale de la voie générale

Sujet n°1

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 `caractere`, un caractère, et `mot`, une chaîne de caractères, et qui renvoie le nombre d'occurrences de `caractere` dans `mot`, c'est-à-dire le nombre de fois où `caractere` apparaît dans `mot`.

Exemples :

```
>>> recherche('e', "sciences")
2
>>> recherche('i',"mississippi")
4
>>> recherche('a',"mississippi")
0
```

EXERCICE 2 (4 points)

On s'intéresse à un algorithme récursif qui permet de rendre la monnaie à partir d'une liste donnée de valeurs de pièces et de billets - le système monétaire est donné sous forme d'une liste `pieces=[100, 50, 20, 10, 5, 2, 1]` - (on supposera qu'il n'y a pas de limitation quant à leur nombre), on cherche à donner la liste de pièces à rendre pour une somme donnée en argument.

Compléter le code Python ci-dessous de la fonction `rendu_glouton` qui implémente cet algorithme et renvoie la liste des pièces à rendre

```
Pieces = [100,50,20,10,5,2,1]

def rendu_glouton(arendre, solution=[], i=0):
       if arendre == 0:
       return ...
    p = pieces[i]
    if p <= ... :
        solution.append(...)
        return rendu_glouton(arendre - p, solution, i)
    else :
        return rendu_glouton(arendre, solution, ...)
```

On devra obtenir :

```
>>>rendu_glouton_r(68,[],0)
[50, 10, 5, 2, 1]
>>>rendu_glouton_r(291,[],0)
[100, 100, 50, 20, 20, 1]
```

```python
from doctest import testmod


def recherche(caractere: str, mot: str) -> int:
    """
    Exemples et tests:
    >>> recherche('e', "sciences")
    2
    >>> recherche('i', "mississippi")
    4
    >>> recherche('a', "mississippi")
    0
    """
    total = 0
    n_mot = len(mot)
    for i in range(n_mot):
        carac_courant = mot[i]
        if carac_courant == caractere:
            total = total + 1
    return total


assert recherche('e', "sciences") == 2
assert recherche('i',"mississippi") == 4
assert recherche('a',"mississippi") == 0

testmod()
```

```python
from doctest import testmod

pieces = [100, 50, 20, 10, 5, 2, 1]

def rendu_glouton(arendre, solution=[], i=0):
    """ Calcul du rendu de monnaie avec un algorithme récursif.

    Args:
        arendre (int): valeur dont on cherche à rendre la monnaie
        solution (list, optional): solution trouvée jusqu'à présent. Defaults to [].
        i (int, optional): rang de la pièce courante. Defaults to 0.

    Returns:
        list: tableau de somme d'argent, pièce par pièce

    Tests et Exemples:
    >>> rendu_glouton(68, [], 0)
    [50, 10, 5, 2, 1]
    >>> rendu_glouton(291, [], 0)
    [100, 100, 50, 20, 20, 1]
    """
    if arendre == 0:
        # cas de base de l'algo récursif
        return solution

    p = pieces[i]
    if p <= arendre :
        # il est possible d'utiliser la pièce courante p
        solution.append(p)
        return rendu_glouton(arendre - p, solution, i)
    else :
        # la pièce courante p est trop grande, il faut changer de pièce
        return rendu_glouton(arendre, solution, i + 1)


assert rendu_glouton(68, [], 0) == [50, 10, 5, 2, 1]
assert rendu_glouton(291,[],0) == [100, 100, 50, 20, 20, 1]

testmod()
```