BACCALAUREAT

SESSION 2021

Épreuve de l'enseignement de spécialité

NUMERIQUE et SCIENCES INFORMATIQUES

Partie pratique

Classe Terminale de la voie générale

Sujet n°8

DUREE DE L'EPREUVE : 1 heure

Le sujet comporte 2 pages numérotées de 1 / 2 à 2 / 2
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

- VOIR MODIFICATIONS -

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

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

modif.py

```
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, ...)


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