BACCALAURÉAT

SESSION 2023

Épreuve de l'enseignement de spécialité

NUMÉRIQUE et SCIENCES INFORMATIQUES

Partie pratique

Classe Terminale de la voie générale

Sujet n°29

DURÉE DE L'ÉPREUVE : 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.

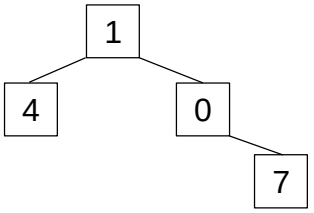
EXERCICE 1 (4 points)

Un arbre binaire est implémenté par la classe `Arbre` donnée ci-dessous.
Les attributs `fg` et `fd` prennent pour valeurs des instances de la classe `Arbre` ou `None`.

```
class Arbre:
    def __init__(self, etiquette):
        self.v = etiquette
        self.fg = None
        self.fd = None
```

L'arbre ci-contre sera donc implémenté de la manière suivante :

```
a=Arbre(1)
a.fg=Arbre(4)
a.fd=Arbre(0)
a.fd.fd=Arbre(7)
```

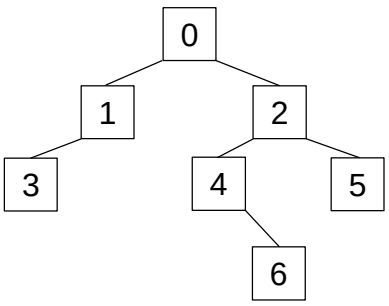


Écrire une fonction récursive `taille` prenant en paramètre une instance `a` de la classe `Arbre` et qui renvoie la taille de l'arbre que cette instance implémente.

Écrire de même une fonction récursive `hauteur` prenant en paramètre une instance `a` de la classe `Arbre` et qui renvoie la hauteur de l'arbre que cette instance implémente.

Si un arbre a un seul nœud, sa taille et sa hauteur sont égales à 1.
S'il est vide, sa taille et sa hauteur sont égales à 0.

Tester les deux fonctions sur l'arbre représenté ci-dessous :



EXERCICE 2 (4 points)

La méthode `insert` de la classe `list` permet d'insérer un élément dans une liste à un indice donné.

Le but de cet exercice est, *sans utiliser cette méthode*, d'écrire une fonction `ajoute` réalisant cette insertion en produisant une nouvelle liste.

Cette fonction `ajoute` prend en paramètres trois variables `indice`, `element` et `liste` et renvoie une liste `L` dans laquelle les éléments sont ceux de la liste `liste` avec, en plus, l'élément `element` à l'indice `indice`.
On considère que les variables `indice` et `element` sont des entiers positifs et que les éléments de `liste` sont également des entiers positifs.
Les éléments de la liste `liste`, dont les indices sont supérieurs ou égaux à `indice` apparaissent décalés vers la droite dans la liste `L`.
Si `indice` est supérieur ou égal au nombre d'éléments de la liste `liste`, l'élément `element` est ajouté dans `L` après tous les éléments de la liste `liste`.

Exemple :

```
>>> ajoute(1, 4, [7, 8, 9])
[7, 4, 8, 9]
>>> ajoute(3, 4, [7, 8, 9])
[7, 8, 9, 4]
>>> ajoute(4, 4, [7, 8, 9])
[7, 8, 9, 4]
```

Compléter et tester le code ci-dessous :

```
def ajoute(indice, element, liste):
    nbre_elts = len(liste)
    L = [0 for i in range(nbre_elts + 1)]
    if ...:
        for i in range(indice):
            L[i] = ...
        L[...] = ...
        for i in range(indice + 1, nbre_elts + 1):
            L[i] = ...
    else:
        for i in range(nbre_elts):
            L[i] = ...
        L[...] = ...
    return L
```