

# **BACCALAURÉAT**

**SESSION 2023**

---

**Épreuve de l'enseignement de spécialité**

## **NUMÉRIQUE et SCIENCES INFORMATIQUES**

**Partie pratique**

**Classe Terminale de la voie générale**

---

**Sujet n°25**

---

**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)

Écrire une fonction `enumere` qui prend en paramètre une liste  $L$  et renvoie un dictionnaire  $d$  dont les clés sont les éléments de  $L$  avec pour valeur associée la liste des indices de l'élément dans la liste  $L$ .

Exemple :

```
>>> enumere([1, 1, 2, 3, 2, 1])  
{1: [0, 1, 5], 2: [2, 4], 3: [3]}
```

## EXERCICE 2 (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`.

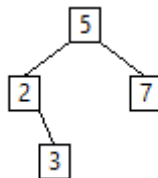
```
class Arbre:
    def __init__(self, etiquette):
        self.v = etiquette
        self.fg = None
        self.fd = None

def parcours(arbre, liste):
    if arbre != None:
        parcours(arbre.fg, liste)
        liste.append(arbre.v)
        parcours(arbre.fd, liste)
    return liste
```

La fonction récursive `parcours` renvoie la liste des étiquettes des nœuds de l'arbre implémenté par l'instance `arbre` dans l'ordre du parcours en profondeur infixe à partir d'une liste vide passée en argument.

Compléter le code de la fonction `insere` qui insère un nœud d'étiquette `cle` en feuille de l'arbre implémenté par l'instance `arbre` selon la spécification indiquée et de façon que l'arbre ainsi complété soit encore un arbre binaire de recherche.

Tester ensuite ce code en utilisant la fonction `parcours` et en insérant successivement des nœuds d'étiquette 1, 4, 6 et 8 dans l'arbre binaire de recherche représenté ci-dessous :



```
def insere(arbre, cle):
    """ arbre est une instance de la classe Arbre qui implémente
        un arbre binaire de recherche.
    """
    if ...:
        if ...:
            insere(arbre.fg, cle)
        else:
            arbre.fg = Arbre(cle)
    else:
        if ...:
            insere(arbre.fd, cle)
        else:
            arbre.fd = Arbre(cle)
```