BACCALAURÉAT

SESSION 2023

Épreuve de l'enseignement de spécialité

NUMÉRIQUE et SCIENCES INFORMATIQUES

Partie pratique

Classe Terminale de la voie générale

Sujet n°33

DURÉE DE L'ÉPREUVE : 1 heure

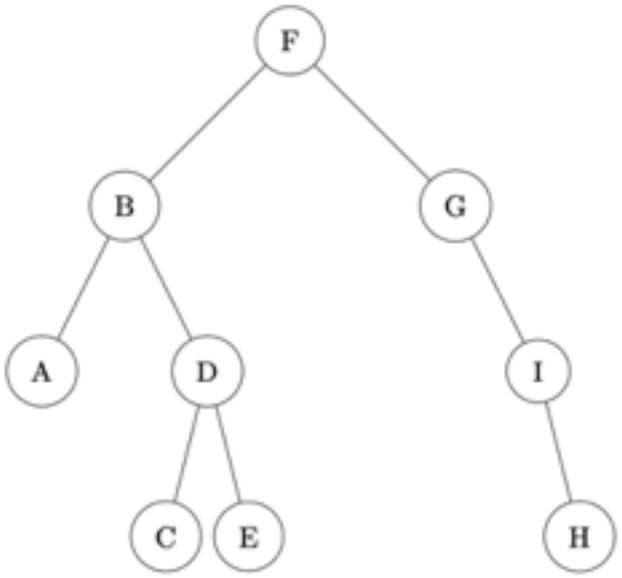
Le sujet comporte 3 pages numérotées de 1 / 3 à 3 / 3
Dès que le sujet vous est remis, assurez-vous qu'il est complet.

Le candidat doit traiter les 2 exercices.

EXERCICE 1 (4 points)

Un arbre binaire de caractères est stocké sous la forme d'un dictionnaire où les clés sont les caractères des nœuds de l'arbre et les valeurs, pour chaque clé, la liste des caractères des fils gauche et droit du nœud.

Par exemple, l'arbre



est stocké dans

```
a = {'F':['B','G'], 'B':['A','D'], 'A':['',''], 'D':['C','E'], \
     'C':['',''], 'E':['',''], 'G':['','I'], 'I':['','H'], \
     'H':['','']}
```

Écrire une fonction récursive `taille` prenant en paramètres un arbre binaire `arbre` sous la forme d'un dictionnaire et un caractère `lettre` qui est la valeur du sommet de l'arbre, et qui renvoie la taille de l'arbre, à savoir le nombre total de nœud.

On observe que, par exemple, `arbre[lettre][0]`, respectivement `arbre[lettre][1]`, permet d'atteindre la clé du sous-arbre gauche, respectivement droit, de l'arbre `arbre` de sommet `lettre`.

Exemple :

```
>>> taille(a, 'F')
9
```

EXERCICE 2 (4 points)

On considère l'algorithme de tri de tableau suivant : à chaque étape, on parcourt le sous-tableau des éléments non rangés et on place le plus petit élément en première position de ce sous-tableau.

Exemple avec le tableau :

```
t = [41, 55, 21, 18, 12, 6, 25]
```

Etape 1 : on parcourt tous les éléments du tableau, on permute le plus petit élément avec le premier. Le tableau devient

```
t = [6, 55, 21, 18, 12, 41, 25]
```

Etape 2 : on parcourt tous les éléments **sauf le premier**, on permute le plus petit élément trouvé avec le second. Le tableau devient :

```
t = [6, 12, 21, 18, 55, 41, 25]
```

Et ainsi de suite.

La code de la fonction `tri_selection` qui implémente cet algorithme est donné ci-dessous.

```
def tri_selection(tab):
    N = len(tab)
    for k in range(...):
        imin = ...
        for i in range(... , N):
            if tab[i] < ... :
                imin = i
        ... , tab[imin] = tab[imin] , ...
```

Compléter le code de cette fonction de façon à obtenir :

```
>>> liste = [41, 55, 21, 18, 12, 6, 25]
>>> tri_selection(liste)
>>> liste
[6, 12, 18, 21, 25, 41, 55]
```

On rappelle que l'instruction

```
a, b = b, a
```

échange les contenus de `a` et de `b`.