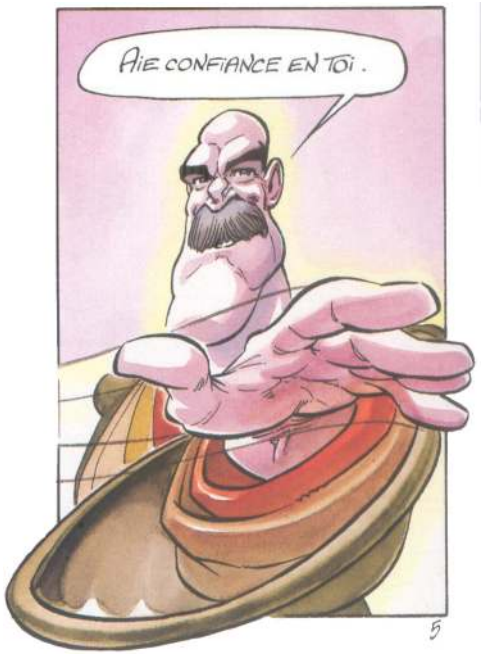
Algorithmique
Contrôle n° 2 (C2)

Info-sup S2
Epita

31 mars 2021 - 9 : 30

Consignes (à lire) :

- Vous devez répondre sur **les feuilles de réponses prévues à cet effet**.
  - Aucune autre feuille ne sera ramassée (gardez vos brouillons pour vous).
  - Répondez dans les espaces prévus, **les réponses en dehors ne seront pas corrigées** : utilisez des brouillons !
  - Ne séparez pas les feuilles à moins de pouvoir les ré-agrafer pour les rendre.
  - Aucune réponse au crayon de papier ne sera corrigée.
- La présentation est notée en moins, c'est à dire que vous êtes noté sur 20 et que les points de présentation (2 au maximum) sont retirés de cette note.
- **Le code :**
  - Tout code doit être écrit dans le langage `Python` (pas de C, Caml, Algo ou autre).
  - **Tout code `Python` non indenté ne sera pas corrigé.**
  - Tout ce dont vous avez besoin (fonctions, méthodes) est indiqué en **annexe** !
- Durée : 2h00



Exercice 1 (Arbre Binaire : Ordres – *2 points*)

Soit un arbre binaire B dont les traitements *préfixe* et *infixe* du parcours profondeur main gauche affichent les séquences suivantes :

préfixe : `A B D H I E C F G J`

infixe : `H D I B E A F C J G`

1. Représenter graphiquement l'arbre B correspondant à ces deux séquences.
2. Donner la séquence affichée lors du traitement *suffixe* du parcours profondeur main gauche de l'arbre binaire B.

Exercice 2 (ABR : insertions – *3 points*)

On veut créer un arbre binaire de recherche par insertions successives, à partir d'un arbre vide, des valeurs `U, N, J, O, L, I, A, B, R`.

Donner le résultat (dessiner l'arbre final) lorsque ces valeurs sont ajoutées :

1. en feuille ;
2. en racine.

Exercice 3 (Matrice triée – *5 points*)

Définition :

Une matrice est *triée* en ordre croissant si :

— chaque ligne de la matrice est triée en ordre croissant

— tous les éléments de chaque ligne de la matrice (excepté la première) sont strictement supérieurs à tous ceux de la ligne précédente.

1. Écrire la fonction `list_sorted(L, n)` qui vérifie si la liste non vide `L` de longueur `n` est triée en ordre croissant.
2. Écrire la fonction `matrix_sorted(M)` qui vérifie si la matrice `M` est *triée* en ordre croissant. La matrice `M` est supposée non vide.

Exemples d'applications :

```
>>> matrix_sorted([[1,2,3]])
True
>>> matrix_sorted([[1,2,3], [40,5,6]])
False
>>> matrix_sorted([[1,2,3], [7,8,9], [4,5,6]])
False
>>> matrix_sorted([[1,2,3], [4,5,6], [7,8,9]])
True
>>> matrix_sorted([[1,-1,3], [4,5,6], [7,8,9]])
False
```

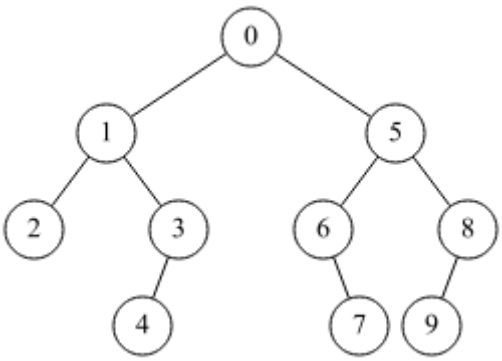


FIGURE 1 – Arbre binaire B

Exercice 4 (Largeur moyenne pondérée – *6 points*)

Écrire la fonction `get_average(B)` qui prend en paramètre un arbre binaire `B` et qui construit et renvoie une liste `L` telle que :

- `len(L)` = nombre de niveaux de l'arbre `B`
- `L[i]` = somme des clés des nœuds du niveau `i` divisée par le nombre de noeuds du niveau `i`

Exemple d'application sur l'arbre de la figure 1 :

```
>>> get_average(B)
[0.0, 3.0, 4.75, 6.666666666666667]
```

Exercice 5 (Recherche du numéro hiérarchique – *4 points*)

Écrire la fonction `search_hier(B, x)` qui prend en paramètre un arbre binaire `B` et une valeur `x` et qui retourne le numéro en ordre hiérarchique de `x` s'il est présent dans B, `None` sinon. Les clés de l'arbre binaire B sont supposées distinctes.

Exemples d'applications sur l'arbre de la figure 1 :

```
>>> print(search_hier(B, 10))
None
>>> search_hier(B, 3)
5
>>> search_hier(B, 0)
1
>>> search_hier(B, 7)
13
```

Annexes

Les arbres binaires

Les arbres binaires manipulés ici sont les mêmes qu'en td.

- L'arbre vide est `None`
- L'arbre non vide est (une référence sur) un objet de la class `BinTree` avec 3 attributs : `key`, `left`, `right`.
  - `B` : classe `BinTree`
  - `B.key` : contenu du nœud racine
  - `B.left` : le sous-arbre gauche
  - `B.right` : le sous-arbre droit

Annexe : Fonctions et méthodes autorisées

`List`

Vous pouvez utiliser la méthode `append` et la fonction `len` sur les listes ainsi que la fonction `range` :

```
>>> L = []

>>> for i in range(5):
        L.append(i)

>>> L
[0, 1, 2, 3, 4]

>>> len(L)
5

>>> for i in range(5, 10):
        L.append(i)
>>> L
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Aucun opérateur n'est autorisé sur les listes (`+`, `*`, `==` ...).

Files

Les méthodes de la classe `Queue`, que l'on suppose importée :

- `Queue()` retourne une nouvelle file ;
- q`.enqueue(`e`)` enfile e dans q ;
- q`.dequeue()` supprime et retourne le premier élément de q ;
- q`.isempty()` teste si q est vide.

Vos fonctions

Vous pouvez également écrire vos propres fonctions, dans ce cas vous devez donnez leurs spécifications : on doit savoir ce qu'elles font.
Dans tous les cas, la dernière fonction écrite doit être celle qui répond à la question.