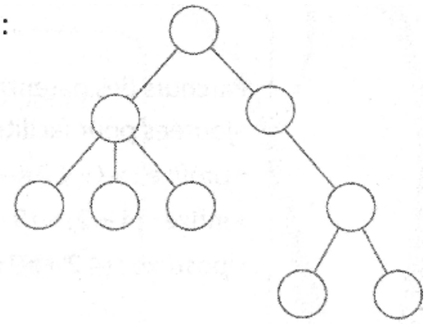
SDD_Arbres_Exercices_Partie_I

Par convention, la racine a une hauteur de zéro dans les exercices de 1 à 6.

Exercice 1 : QCM, vocabulaire des arbres (une seule réponse possible)

On considère l'arbre non étiqueté ci-contre :



1. Quelle est sa hauteur ?

☐ a. 9 ☐ b. 3
☐ c. 2 ☐ d. 8

2. Quelle est sa taille ?

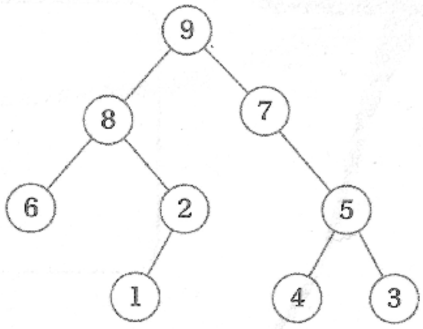
☐ a. 9
☐ b. 8
☐ c. 4

3. Quel est l'arité maximale parmi les nœuds de l'arbre ?

☐ a. 1 ☐ b. 2
☐ c. 3 ☐ d. 4

Exercice 2 : QCM, vocabulaire du parcours d'arbres (une seule réponse possible)

On considère l'arbre binaire ci-contre étiqueté par des entiers.



1. Dans quel ordre seront examinés les nœuds lors d'un parcours en largeur ?

☐ a. 6–1–2–8–4–3–5–7–9
☐ b. 9–8–7–6–2–5–1–4–3
☐ c. 6–8–1–2–9–7–4–5–3
☐ d. 9–8–6–2–1–7–5–4–3

2. Dans quel ordre seront examinés les nœuds lors d'un parcours préfixe ?

☐ a. 6–1–2–8–4–3–5–7–9 ☐ c. 6–8–1–2–9–7–4–5–3
☐ b. 9–8–7–6–2–5–1–4–3 ☐ d. 9–8–6–2–1–7–5–4–3

3. Dans quel ordre seront examinés les nœuds lors d'un parcours infixe ?

☐ a. 6–1–2–8–4–3–5–7–9 ☐ c. 6–8–1–2–9–7–4–5–3
☐ b. 9–8–7–6–2–5–1–4–3 ☐ d. 9–8–6–2–1–7–5–4–3

4. Dans quel ordre seront examinés les nœuds lors d'un parcours postfixe ?

☐ a. 6–1–2–8–4–3–5–7–9 ☐ c. 6–8–1–2–9–7–4–5–3
☐ b. 9–8–7–6–2–5–1–4–3 ☐ d. 9–8–6–2–1–7–5–4–3

Exercice 3 : QCM, arbre d'expression arithmétique (plusieurs réponses possibles)

Dans un arbre binaire représentant une expression arithmétique, que peut-on affirmer ?

a) Chaque nœud interne a exactement deux fils.
b) Chaque nœud a un ou deux fils.
c) Les feuilles contiennent des nombres.
d) Les nombres sont disposés sur des feuilles ou des nœuds internes.
e) Les opérateurs ne peuvent pas être sur des feuilles.

<u>Exercice 4</u> : QCM, ABR (une seule réponse possible)
Dans un ABR, la plus petite valeur se trouve forcément :

a) Sur la feuille la plus à gauche.
b) Sur la racine
c) Sur la feuille la plus à droite.

<u>Exercice 5</u> : Construire un ABR
1/ **Dessiner** un ABR contenant les valeurs 11, 13, 14, 15, 17, 18, 19 tel que la hauteur de cet arbre soit 2.
2/ Avec les même valeurs que dans le 1/, **dessiner** un ABR tel que le sous-arbre gauche contienne 4 nœuds et le sous-arbre droit 2 nœuds.

<u>Exercice 6</u> : Reconstruire un arbre
Un arbre binaire est étiqueté avec des lettres. Un parcours <u>préfixe</u> de l'arbre donne : ALORHGIMET. Un parcours <u>infixe</u> donne : OLHRAMIEGT.

a) **Reconstruire** l'arbre binaire ayant produit ces deux résultats.
b) Qu'obtient-on en faisant un **parcours en largeur** d'abord ? Et en faisant un **parcours postfixe** ?

<u>Exercice 7</u> : ABR
Dans cet exercice, les arbres binaires de recherche ne peuvent pas comporter plusieurs fois la même clé. De plus, un arbre binaire de recherche limité à un nœud a une hauteur de 1. On considère l'arbre binaire de recherche représenté ci-dessous (figure 1), où val représente un entier :

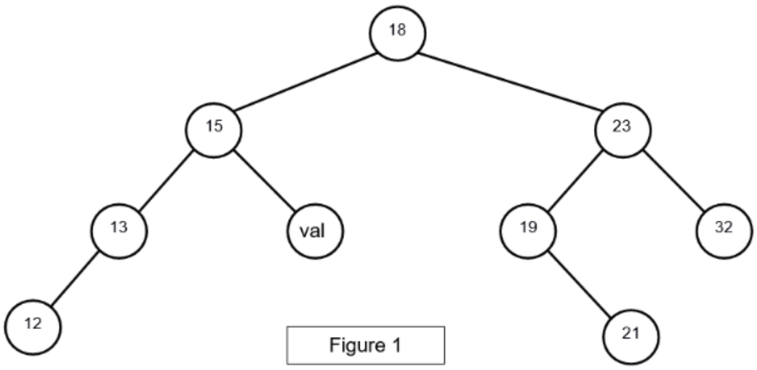


Figure 1

1/

a) **Donner** le nombre de feuilles de cet arbre et préciser leur valeur (étiquette).
b) **Donner** le sous arbre-gauche du nœud 23.
c) **Donner** la hauteur et la taille de l'arbre.
d) **Donner** les valeurs entières possibles de val pour cet arbre binaire de recherche.

On suppose, pour la suite de cet exercice, que *val* est égal à 16.

2/ On rappelle qu'un parcours infixe depuis un nœud consiste, dans l'ordre, à faire un parcours infixe sur le sous arbre-gauche, afficher le nœud puis faire un parcours infixe sur le sous-arbre droit.

Dans le cas d'un parcours suffixe (ou postfixe), on fait un parcours suffixe sur le sous-arbre gauche puis un parcours suffixe sur le sous-arbre droit, avant d'afficher le nœud.

a) **Donner** les valeurs d'affichage des nœuds dans le cas du parcours infixe de l'arbre.
b) **Donner** les valeurs d'affichage des nœuds dans le cas du parcours suffixe de l'arbre.

<u>Exercice 8</u> (*) : Ajout d'un élément dans un ABR

Il y a plusieurs façons d'implémenter un ABR. On peut utiliser une structure de classe qui peut servir un arbre binaire quelconque.

Un nœud a deux enfants au maximum (pouvant être `None`) et, excepté à la racine à la racine, possède un parent unique. Chaque nœud possède une valeur. On crée donc **une classe `Node`** avec quatre attributs :

- Une valeur.
- Un parent.
- Un enfant à gauche.
- Un enfant à droite.

```python
# Définition de la classe `Node`
class Node :
    def __init__(self, val) :
        self.value = val
        self.parent = None  # Valeur par défaut
        self.left = None    # Valeur par défaut
        self.right = None   # Valeur par défaut


    # Surcharge de l'instruction `print`. Affiche la valeur du noeud
    def __str__(self) :
        return str(self.value)


    #######################################################
    ################# A COMPLETER #########################
    #######################################################
    def add(self, val) :
        pass

    #######################################################

tree = Node(15) # Racine de l'arbre
print(tree)     # Affiche la valeur de l'objet `tree`
```

1/ Sur Jupyter, **recopier** le programme précédent. Vérifier que l'on obtient bien `15` à l'exécution.

2/ (*) **Ecrire** la méthode `***add(self, val)***` qui permet d'ajouter une valeur à l'arbre en tenant compte des propriétés d'un ABR (on supprimera l'instruction `*pass*`), les <u>valeurs, assimilées à des clefs, étant toutes différentes</u>.

<u>Aide</u> : procéder par récursivité en étudiant deux cas :

- si la valeur ajoutée est inférieure à la valeur du nœud parent, on l'ajoute au sous-arbre gauche,
- si la valeur ajoutée est supérieure à la valeur du nœud parent, on l'ajoute au sous-arbre droit.
- Dans chaque cas, si le sous-arbre (à gauche ou droite) est `*None*`, on crée alors un nouveau nœud et le nœud courant devient son parent.