Exercice 3 (4 points)

Cette exercice traite des piles, des arbres et de l'algorithmique.

Dans cet exercice, on s'intéresse à la notation polonaise inversée (NPI) d'une expression mathématique. Dans cette notation, l'opérateur est placé après les nombres sur lesquels il s'applique. On se limitera aux expressions faisant intervenir des nombres entiers et les quatre opérateurs : $+, -, \times, /$

Par exemple, l'expression $4 \times (5 + 7)$ s'écrit, en NPI : $4\ 5\ 7\ +\ \times$

L'évaluation de l'expression $12 / 2 - 4 + 5 \times 3$, écrite en NPI : $12\ 2\ /\ 4\ -\ 5\ 3\ \times\ +$ est détaillée ci-dessous. Cette expression s'évalue à 17 de la manière suivante :

- lorsqu'on rencontre un premier opérateur $(+, -, /, \times)$, on évalue l'opération avec les deux nombres situés juste avant cet opérateur :

$$\underbrace{12\ 2\ /}_{12\,/\,2\,=\,6}\ 4\ -\ 5\ 3\ \times\ +$$

- on remplace cette opération par le résultat :

$$6\ 4\ -\ 5\ 3\ \times\ +$$

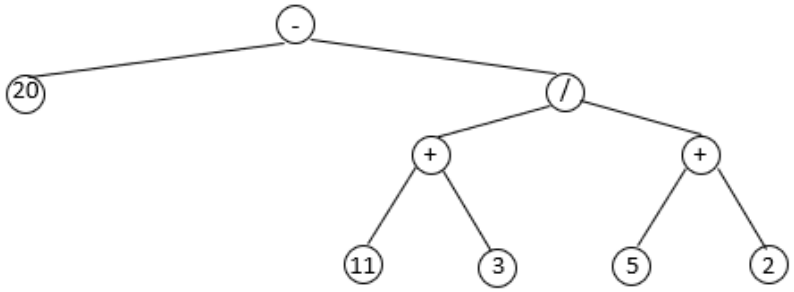
- on continue la lecture de l'expression :

$\underbrace{6\ 4\ -}_{6\,-\,4\,=\,2}\ 5\ 3\ \times\ +$ devient $2\ \underbrace{5\ 3\ \times}_{5\times3=15}\ +$ devient $2\ 15\ +$ qui vaut 17.

1. Donner la valeur de l'expression suivante écrite en NPI :

$$15\ 5\ -\ 4\ 12\ +\ \times$$

2. On donne l'arbre binaire suivant :



Indiquer, parmi les quatre différents parcours d'arbre ci-dessous celui qui permet d'obtenir la succession des symboles correspondant à la notation NPI suivante :

$$20\ 11\ 3\ +\ 5\ 2\ +\ /\ -$$

- un parcours en largeur ;
- un parcours en profondeur préfixe ;
- un parcours en profondeur infixe ;
- un parcours en profondeur postfixe (ou suffixe).

3. On utilisera une liste de symboles pour écrire la notation polonaise inversée d'une expression mathématique. Par exemple, $6\ 2\ 3\ +\ \times\ 94\ 1\ -\ +$ sera représentée par la liste :

[6, 2, 3, +, x, 94, 1, -, +]

On considère à présent l'algorithme ci-dessous, écrit en pseudo-code, qui reçoit une liste de symboles correspondant à la notation polonaise inversée d'une expression et renvoie l'arbre binaire associé.

```
Algorithme créer_arbre(liste_symboles)
P ← créer une pile vide
pour chaque élément de liste_symboles :
    si élément est un entier :
        a ← construire l'arbre de racine élément et n'ayant pas de sous-arbre droit ni gauche
    sinon : # élément est le symbole d'un opérateur
        a_droit ← dépiler P
        a_gauche ← dépiler P
        a ← construire l'arbre de racine élément,
                ayant pour sous-arbre gauche a_gauche et pour sous-arbre droit a_droit
    empiler a dans P.
a ← dépiler P
renvoyer a
```

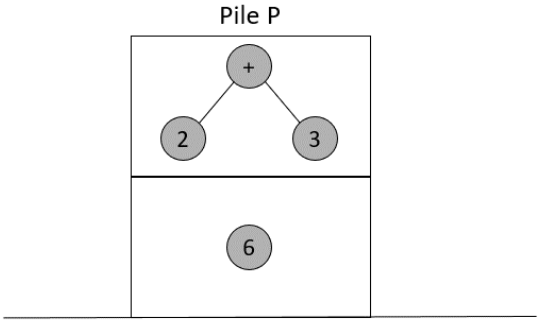
L'algorithme précédent nécessite l'utilisation d'une pile, dont les éléments sont des arbres.

On précise que :

- l'action empiler *a* dans *P* insère l'élément *a* en haut de la pile *P* ;
- l'action dépiler *P* renvoie la valeur de l'élément en haut de la pile *P* et le supprime de la pile.

a. Des deux acronymes LIFO et FIFO, indiquer lequel permet de décrire, de manière générale, la structure de pile. Donner la signification des quatre lettres qui composent cet acronyme.

b. On applique l'algorithme précédent sur la liste [6, 2, 3, +, x, 94, 1, -, +].
À la fin de l'exécution du quatrième tour de la boucle pour, l'état de la pile est le suivant :



Indiquer le nombre d'élément(s) contenu(s) dans la pile *P* et dessiner l'état de la pile, à la fin de l'exécution du cinquième tour de la boucle pour.

c. Dessiner l'arbre binaire représentant l'expression mathématique dont la notation polonaise inversée est la liste [6, 2, 3, +, x, 94, 1, -, +].

4. On souhaite écrire une fonction en langage Python qui permet d'évaluer une expression mathématique écrite avec la notation polonaise inversée NPI. On suppose qu'une telle expression est représentée par un arbre binaire obtenu à l'aide de l'algorithme précédent.

On dispose pour cela de quatre fonctions :

- `est_vide(arb)` qui renvoie `True` si l'arbre binaire `arb` est vide, `False` sinon ;
- `racine(arb)` qui renvoie la valeur de la racine de l'arbre binaire `arb` ;
- `gauche(arb)` qui renvoie le sous-arbre gauche de l'arbre binaire `arb` ;
- `droit(arb)` qui renvoie le sous-arbre droit de l'arbre binaire `arb`.

La fonction `evaluer` donnée ci-dessous est écrite en Python.

Cette fonction prend en paramètre un arbre binaire `arb` représentant une expression mathématique écrite en NPI et renvoie la valeur de cette expression.

Numéro de lignes	Fonction `evaluer`
1	`def evaluer(arb):`
2	`    if est_vide(gauche(arb)) and à compléter (instruction 1):`
3	`        res = à compléter (instruction 2)`
4	`    elif à compléter (instruction 2) == "+":`
5	`        res = evaluer(à compléter (instruction 3)) + à compléter (instruction 4)`
6	`    elif à compléter (instruction 2) == "-":`
7	`        res = evaluer(à compléter (instruction 3)) - à compléter (instruction 4)`
8	`    elif à compléter (instruction 2) == "*":`
9	`        res = evaluer(à compléter (instruction 3)) * à compléter (instruction 4)`
10	`    else:`
11	`        res = evaluer(à compléter (instruction 3)) / à compléter (instruction 4)`
12	`    return res`

Quatre instructions seulement permettent de compléter ce code : ***instruction 1***, ***instruction 2***, ***instruction 3*** et ***instruction 4***.

Écrire sur la copie le code de chacune de ces quatre instructions.