

EXERCICE 1 (4 points)

Le codage par différence (*delta encoding* en anglais) permet de compresser un tableau de données en indiquant pour chaque donnée, sa différence avec la précédente (plutôt que la donnée elle-même). On se retrouve alors avec un tableau de données assez petites nécessitant moins de place en mémoire. Cette méthode se révèle efficace lorsque les valeurs consécutives sont proches.

Programmer la fonction `delta` qui prend en paramètre un tableau non vide de nombres entiers et qui renvoie un tableau contenant les valeurs entières compressées à l'aide cette technique.

Exemples :

```
>>> delta([1000, 800, 802, 1000, 1003])
[1000, -200, 2, 198, 3]
>>> delta([42])
42
```

EXERCICE 2 (4 points)

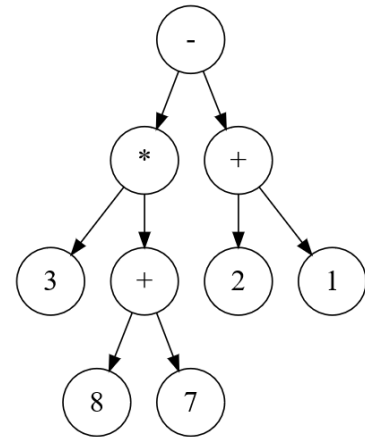
Une expression arithmétique ne comportant que les quatre opérations $+$, $-$, \times , \div peut être représentée sous forme d'arbre binaire. Les nœuds internes sont des opérateurs et les feuilles sont des nombres. Dans un tel arbre, la disposition des nœuds joue le rôle des parenthèses que nous connaissons bien.

En parcourant en profondeur infixe l'arbre binaire ci-contre, on retrouve l'expression notée habituellement :

$$3 \times (8 + 7) - (2 + 1).$$

La classe `Noeud` ci-après permet d'implémenter une structure d'arbre binaire.

Compléter la fonction récursive `expression_infixe` qui prend en paramètre un objet de la classe `Noeud` et qui renvoie l'expression arithmétique représentée par l'arbre binaire passé en paramètre, sous forme d'une chaîne de caractères contenant des parenthèses.



Résultat attendu avec l'arbre ci-dessus :

```
>>> e = Noeud(Noeud(Noeud(None, 3, None), '*', Noeud(Noeud(None, 8, None),
'+', Noeud(None, 7, None))), '-', Noeud(Noeud(None, 2, None), '+',
Noeud(None, 1, None)))

>>> expression_infixe(e)
'((3*(8+7))-(2+1))'
```

```
class Noeud:
```

```

...
Classe implémentant un noeud d'arbre binaire disposant de 3
attributs :
- valeur : la valeur de l'étiquette,
- gauche : le sous-arbre gauche.
- droit : le sous-arbre droit.
...
def __init__(self, g, v, d):
    self.gauche = g
    self.valeur = v
    self.droit = d

def est_une_feuille(self):
    '''Renvoie True si et seulement si le noeud est une feuille'''
    return self.gauche is None and self.droit is None

def expression_infixe(e):
    s = ...
    if e.gauche is not None:
        s = s + expression_infixe(...)
    s = s + ...
    if ... is not None:
        s = s + ...
    if ...:
        return s

    return '(' + s + ')'

```