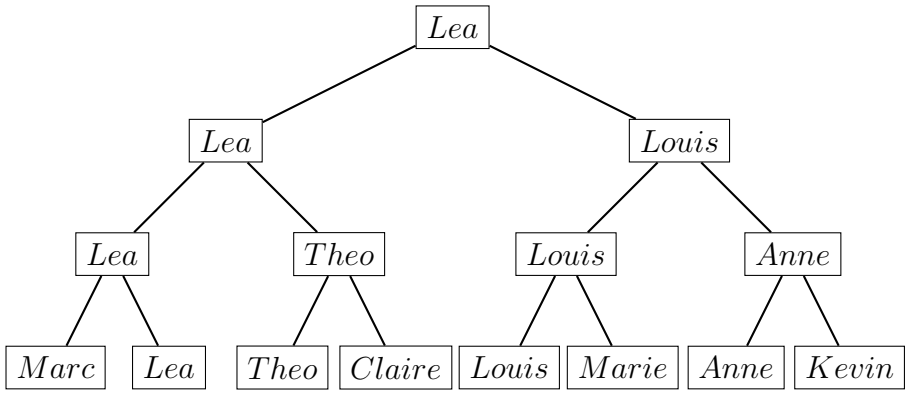
Exercice n° 1

La fédération de badminton souhaite gérer ses compétitions à l'aide d'un logiciel. Pour ce faire, une structure arbre de compétition a été définie récursivement de la façon suivante :
Un arbre de compétition est
- soit l'arbre vide,
- soit un triplet composé d'une chaîne de caractères appelée valeur, d'un arbre de compétition appelé sous-arbre gauche et d'un arbre de compétition appelé sous-arbre droit.

On note **B** l'arbre de compétition représenté ci-dessous.



Pour alléger la représentation d'un arbre de compétition, on ne notera pas les arbres vides.
Cet arbre se lit de la façon suivante :
— Huits participants se sont affrontés : Marc, Lea, Claire, Theo , ...
— Au premier tour : Lea a battu Marc. Theo a battu Claire. ...
— En finale, Lea a battu Louis. Elle est donc vainqueur de la compétition.

Les quatre fonctions suivantes pourront être utilisées :
— La fonction **racine** qui prends en paramètre un arbre de compétition **arb** et renvoie la valeur de la racine.
Exemple : racine(B) vaut Lea
— La fonction **gauche** qui prends en paramètre un arbre de compétition **arb** et renvoie le sous-arbre gauche de **arb**.
— La fonction **droit** qui prends en paramètre un arbre de compétition **arb** et renvoie le sous-arbre droit de **arb**.
— La fonction **est_vide** qui prends en paramètre un arbre de compétition **arb** et renvoie **True** si l'arbre **arb** est vide et **False** sinon.

Pour toutes les questions de l'exercice, on suppose que tous les joueurs d'une même compétition ont un prénom différent.

1. Quelle est la taille de l'arbre **B** ? Combien possède t-il de feuilles ?

2. Proposer une fonction **vainqueur** écrite en Python prenant pour argument un arbre de compétition **arb** ayant au moins un joueur. Cette fonction doit renvoyer le prénom du vainqueur du tournoi.

3. Proposer une fonction **finale** écrite en Python prenant pour argument un arbre de compétition **arb** ayant au moins deux joueurs. Cette fonction doit renvoyer le tableau des prénoms des deux finalistes.

4. Compléter la fonction **liste_joueur** qui prends pour argument un arbre de compétition **arb** et qui renvoie un tableau contenant les participants au tournoi, chaque prénom ne devant figurer qu'une seule fois dans le tableau.

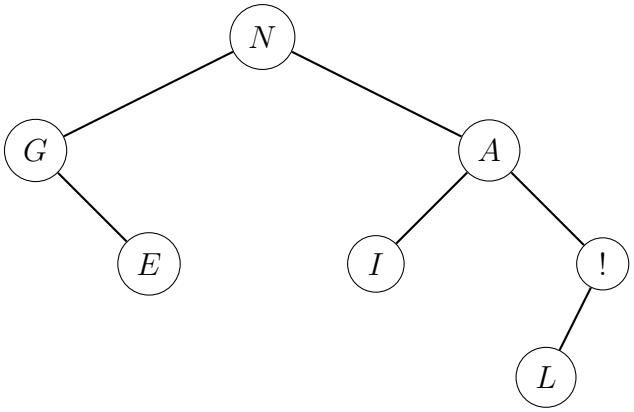
```
def liste_joueur(arb):
    """ arbre_competition -> tableau"""
    if est_vide(arb):
        return ......................
    elif ......................  and ...................... :
        return [racine(arb)]
    else:
        return ...................... +  liste_joueur(droit(arb))
```

L'opération + à la dernière ligne permet de concaténer deux tableaux.
Exemple : ["B", "R" , "A"] +["V","O"] donnera ["B", "R", "A", "V", "O"]

5. Proposer une fonction **occurences** écrite en Python prenant pour arguments un arbre de compétition **arb** et le prénom d'un joueur **prenom**. Cette fonction doit renvoyer le nombre d'occurences (d'apparitions) du **prenom** dans l'arbre **arb**.
Exemple : occurences(B,"Anne") vaut 2.
6. **EN DEDUIRE** une fonction **nb_match** écrite en Python prenant pour arguments un arbre de compétition **arb** et le prénom d'un joueur **prenom**. Cette fonction doit renvoyer le nombre de match disputé par le joueur.
Exemples : nb_match(B,"Anne") vaut 2 et **nb_match(B,"Lea")** vaut 3.

Exercice n° 2 Parcours dans un arbre binaire

1. On considère l'arbre binaire ci-dessous :



   a. Donner un parcours en profondeur infixe.
   b. On a obtenue le parcours suivant : [N,G,A,E,I, !,L]. De quel type de parcours s'agit-il ?
   c. Quel type de parcours a pour algorithme récursif :

```
ParcoursMystere(arbre):
    Si arbre n'est pas vide
        Afficher l'étiquette de arbre
        ParcoursMystere(filsGauche(arbre))
        ParcoursMystere(filsDroit(arbre))
```

2. **a.** Un parcours en largeur utilise t-il une structure de pile ou de file ?
   b. Écrire un algorithme que permet de parcourir un arbre binaire en largeur.