TD 5 : Récursivité (2)

Info1.Algo1

2022-2023 Semestre Impair

Rappels sur les arbres

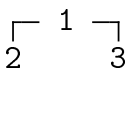
```
def creer_arbre_vide():
  return None

def creer_arbre(r,g,d):
  return r,g,d

def est_vide(arbre):
  return arbre==None
```

```
def racine(arbre):
  return arbre[0]

def gauche(arbre):
  return arbre[1]

def droite(arbre):
  return arbre[2]
```

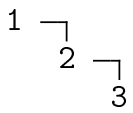
Exercice 1 : impléméntation

Associer à chacun des arbres représentés graphiquement son implémentation sous forme de tuples :

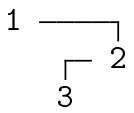
Arbre A



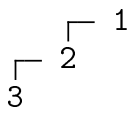
Arbre B



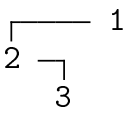
Arbre C



Arbre D



Arbre E



- Arbre 1 : `(1,(2,(3,None,None),None),None)`
- Arbre 2 : `(1,(2,None,None),(3,None,None))`
- Arbre 3 : `(1,(2,None,(3,None,None)),None)`
- Arbre 4 : `(1,None,(2,None,(3,None,None)))`
- Arbre 5 : `(1,None,(2,(3,None,None),None))`

Exercice 2 : appartenance

Écrire la fonction récursive `appartient` qui, accepte en paramètres un arbre et une valeur, et retourne `True` si la valeur est présente dans l'arbre et `False` sinon.

Exercice 3 : somme des valeurs

Écrire la fonction récursive `somme` qui accepte en paramètre un arbre d'entiers et retourne la somme des valeurs présentes dans l'arbre.
Indication : la somme des éléments d'un arbre vide est `0`.

Exercice 4 : feuille gauche

Écrire la fonction récursive `feuille_gauche` qui accepte en paramètre un arbre **non vide** et retourne la valeur associée à la feuille la plus à gauche.

Exercice 5 : sous-arbre par valeur

Écrire une fonction récursive `sous_arbre` qui accepte en paramètres un arbre contenant des valeurs entière ainsi qu'une valeur `v` entière. La fonction retourne le sous-arbre dont la racine est la valeur `v`.

Indications :

- S'il y en a plusieurs, on retourne celui qui est le plus à gauche.
- S'il n'y en a pas, elle renvoie un arbre vide.

Exercice 6 : miroir

Écrire la fonction récursive `miroir` qui accepte en paramètre un arbre et retourne l'arbre obtenu par symétrie verticale.

Exercice 7 : substitution

Écrire la fonction récursive `substituer` qui accepte en paramètres :

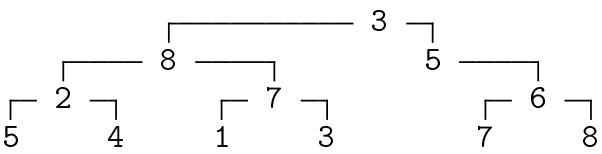
- un arbre
- deux valeurs : `ancienne_valeur` et `nouvelle_valeur`.

La fonction construit et retourne un nouvel arbre dans lequel toute occurrence de `ancienne_valeur` a été remplacée par `nouvelle_valeur`.

Exercice 8 : parcours infixe

Écrire la fonction récursive `parcours_infixe` qui accepte en paramètre un arbre et retourn la liste (de type natif `list`) des éléments s'y trouvant dans l'ordre du **parcours infixe**.
(c'est-à-dire l'ordre **gauche - racine - droite**)

Exemple : pour l'arbre :



la liste retournée est :
`[5,2,4,8,1,7,3,3,5,7,6,8]`

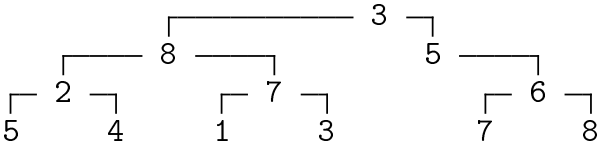
Exercice 9 : cheminement

1) Écrire une fonction récursive `acces_par_chemin` qui accepte en paramètres :

- Un arbre
- Une chaîne de caractère `chemin` (éventuellement vide) constituée uniquement de caractères `'g'` (pour *gauche*) et de `'d'` (pour *droite*).

La fonction accède à l'élément de l'arbre décrit par `chemin` et retourne la valeur rencontrée.

Exemple : si l'arbre est



et si `chemin` vaut ggd alors la valeur retournée est 4

2) Écrire une fonction récursive `chemin_vers_feuille` qui calcule le chemin à suivre depuis la racine d'un arbre pour aboutir à la feuille la plus profonde (la plus à gauche en cas d'égalité).