

Entraînement : Arbres structure récursive

Pour tous les exercices, la grille d'évaluation est la suivante.

Application d'algorithmes.

A (20)	Valeur des fonctions et interprétation correcte
C (11)	Valeurs des fonctions correctes mais mauvaise interprétation en général.
D (8)	Erreurs pour les valeurs dans une des deux fonctions.
E (1)	Les deux sont fausses.

Conception d'algorithmes.

A (20)	L'algorithme répond correctement au problème posé, il est écrit de façon claire et la complexité est optimale.
B (16)	L'algorithme contient quelques erreurs mais reste globalement juste et la complexité est optimale.
C (11)	L'algorithme fonctionne uniquement dans certains cas ou bien la complexité n'est pas optimale.
D (8)	L'algorithme ne fonctionne pas.
E (1)	Algorithme quasi inexistant ou ne répondant pas du tout au problème posé.

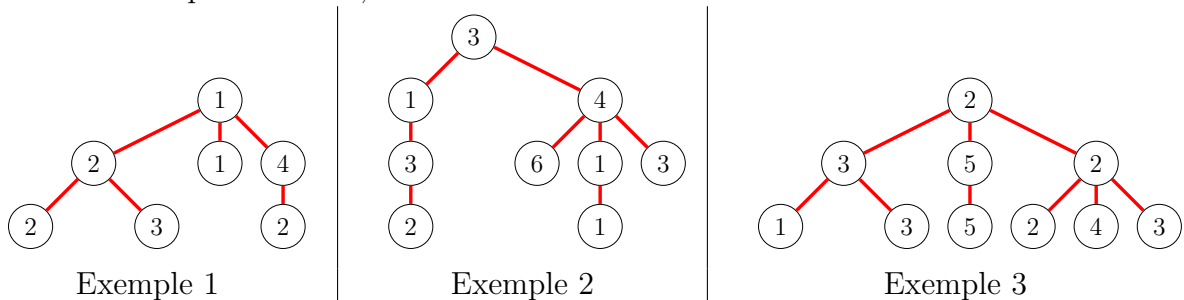
Les questions ♣ sont "bonus" : on ne mettra que *A*, *B* ou *C*, il faut les réussir pour avoir 20 mais elles n'affectent pas la validation du partiel.

Exercice 1.

On définit la structure suivante pour représenter les arbres :

```
Structure Arbre :
    valeur, un entier
    nbFils, un entier
    Fils, un tableau de taille nbFils contenant des Arbres
```

Et les trois exemples suivants,



(1) Donner les valeurs de `mystere1` et `mystere2` sur les 3 exemples d'arbres donnés.

```
Fonction mystere1(Arbre arbre) :
    s = 1
    Pour chaque fils f de arbre :
        s = s + mystere1(f)
    retourner s

Fonction mystere2(Arbre arbre) :
    m = 0
    Pour chaque fils f de arbre :
```

```

    h = mystere2(e)
    Si h > m :
        m = h
    retourner m + 1
Fin fonction

```

- (2) Que calculent **mystere1** et **mystere2** ?
- (3) Donner une fonction qui calcule le nombre de *feuilles* de l'arbre (nombre de nœuds qui n'ont pas de fils). La fonction doit retourner 4 sur l'exemple 1, 4 sur l'exemple 2 et 6 sur l'exemple 3.
- (4) ♣ On dit qu'un arbre est équilibré si toutes les feuilles sont à la même profondeur. Dans les arbres donnés en exemple, seul l'exemple 3 est équilibré (toutes ses feuilles sont à la profondeur 3, pour les exemples 1 et 2, il y a des feuilles aux profondeurs 2 et 3). Proposer un algorithme qui teste si un arbre donné est équilibré. (Note : il y a plusieurs solutions possibles, plus l'algorithme est optimal en terme de complexité, mieux c'est).

Exercice 2 (Arbres Binaires).

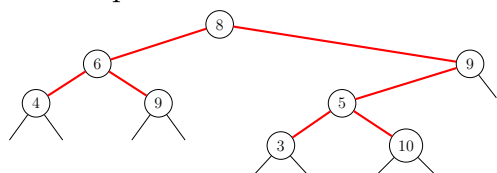
On définit la structure suivante pour représenter les arbres binaires :

```

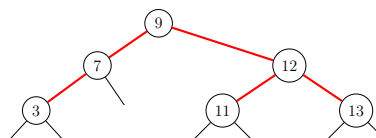
Structure ArbreBinaire :
    valeur , un entier
    filsGauche , un ArbreBinaire
    filsDroit , un ArbreBinaire

```

Si un fils est vide, on considérera qu'il est égal à **None**, sur les exemples, on représente les arbres vides par des arrêtes sans noeud au bout.



Exemple 1



Exemple 2

- (1) Donner un algorithme calculant la hauteur d'un arbre binaire. Sur les exemples, vous devez trouver 4 sur l'Exemple 1 et 3 sur l'Exemple 2.
- (2) Donner les valeurs de la fonction **Strahler** sur les 2 exemples d'arbres donnés.

```

Strahler
Input :
    - A, un arbre binaire ou bien None
Output :
    - Un entier
Procédé :
    Si A = None :
        retourner 0
    s1 <- Strahler(A.filsGauche)
    s2 <- Strahler(A.filsDroit)
    Si s1 = s2 :
        retourner s1+1
    Sinon :
        retourner max(s1, s2)

```

- (3) Donner le résultat de la fonction suivante sur les deux exemples d'arbres donnés. Comment s'appelle ce parcours ?

```
Parcours
Input  :
    - A, un arbre binaire ou bien None
Output :
    - Une liste d'entiers
Procédé :
    Si A = None :
        retourner [] # Liste vide
    L <- [A.valeur] # liste contenant une seule valeur
    L1 <- Parcours(A.filsGauche)
    L2 <- Parcours(A.filsDroit)
    retourner L1 + L + L2 # concaténation de listes
```

- (4) On rappelle qu'un arbre binaire de recherche est un arbre tel que pour chaque noeud, les noeuds de son sous arbre gauche ont des valeurs **plus petites ou égales** et les noeuds de son sous arbre droit ont des valeurs **strictement supérieures**. L'exemple 1 n'est pas un arbre binaire de recherche et l'exemple 2 en est un.
- (a) Écrivez une fonction qui prend en paramètre un arbre binaire de recherche et un entier v et qui retourne Vrai si v est dans l'arbre et Faux sinon.
 - (b) De quoi dépend la complexité de votre algorithme ?
 - (c) Quelle est la particularité d'un parcours infixe dans un arbre binaire de recherche ?
 - (d) ♣ Utilisez cette propriété pour écrire une fonction qui teste si un arbre donné est un arbre binaire de recherche. (Vous pouvez utiliser les fonctions précédentes)