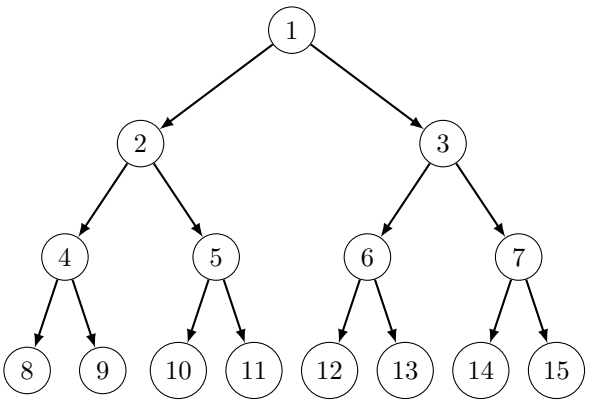
Exercice n° 1

1 Arbre binaire complet



1. Par quel(s) adjectif(s) peut-on qualifier cet arbre ?
2. Donner les caractéristiques de cet arbre :
   — Taille
   — Hauteur (en prenant la profondeur de la racine égale à 0)
   — Nombre de feuilles

   Pour chaque nœud dont on connaît l'étiquette (c'est à dire la clé), comment est déterminée :
   — l'étiquette de la racine du sous-arbre gauche ?
   — l'étiquette de la racine du sous-arbre droit ?

   Écrire une fonction récursive `arbre_complet(hauteur,etiquette_racine)` retournant un arbre complet dont la hauteur et l'étiquette de la racine sont passées en paramètres. Les étiquettes des nœuds de cet arbre seront déterminées automatiquement selon la méthode étudiée précédemment.

   Exemple :

```
>>> complet(2,1)
[1, [2, [4, [], []], [5, [], []]], [3, [6, [], []], [7, [], []]]]
```

Exercice n° 2

1. Écrire une fonction qui vérifie si un arbre binaire est un arbre binaire complet (Tous les nœuds internes ont exactement deux fils et toutes les feuilles ont la même profondeur)
2. Vrai ou Faux ? Un arbre binaire complet est un arbre binaire de hauteur h possédant $2^{h+1} - 1$ nœuds.

2 Arbre miroir

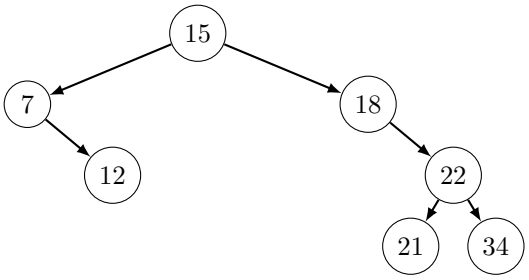
Écrire une fonction récursive miroir retournant le "miroir" (au sens d ?une symétrie par rapport à un axe vertical) de l'arbre passé en paramètre. Exemple :

```
>>> pommier = [1, [2, [4, [], []], [5, [], []]], [3, [6, [], []], [7, [], []]]]
>>> miroir(pommier)
[1, [3, [7, [], []], [6, [], []]], [2, [5, [], []], [4, [], []]]]
```

3 Arbre binaire de recherche

Exercice n° 3

1. L'arbre binaire ci-dessous est-il un arbre de recherche ?



2. Ajouter successivement à l'arbre binaire ci-dessus les valeurs : 9, 14, 16, 23 et 5.

Exercice n° 4

Dessiner des arbres binaires de recherche de hauteur 2, 3, 4, 5 et 6 pour le même ensemble de clefs : $1, 4, 5, 10, 16, 17, 21$

Exercice n° 5

Dessiner tous les arbres binaires de recherche contenant les valeurs 1, 2, 3, 4 et 5 et ayant 3 comme valeur à la racine.

Exercice n° 6

1. On suppose que les entiers compris entre 1 et 1000 sont disposés dans un arbre binaire de recherche, et on souhaite retrouver le nombre 363. Parmi les séquences suivantes, lesquelles ne pourraient pas être la séquence de nœuds parcourus ?

a. 2, 252, 401, 398, 330, 344, 397, 363.

b. 924, 220, 911, 244, 898, 258, 362, 363

c. 925, 202, 911, 240, 912, 245, 363

d. 2, 399, 387, 219, 266, 382, 381, 278, 363

e. 935, 278, 347, 621, 299, 392, 358, 363

2. Écrire un algorithme qui permettrait de faire ce travail de vérification.

Exercice n° 7

Écrire deux fonctions qui retournent respectivement la plus petite et la plus grande valeur contenue dans un arbre binaire de recherche.

Exercice n° 8

Écrire une fonction qui vérifie si un arbre binaire est un arbre binaire de recherche.

Exercice n° 9

Le professeur Bouzin pense avoir découvert une remarquable propriété des arbres binaires de recherche. On suppose que la recherche d'un élément k dans un arbre binaire de recherche se termine sur une feuille. On considère trois ensembles :

— A, les clés situées à gauche du chemin de recherche ;
— B, celles situées le long du chemin de recherche ;
— C, les clés situées à droite du chemin de recherche.

Le professeur Bouzin affirme que quelque soit le triplet (a, b, c) avec $a \in A$, $b \in B$ et $c \in C$ on a $a \leqslant b \leqslant c$.
Donner un contre-exemple aussi petit que possible.