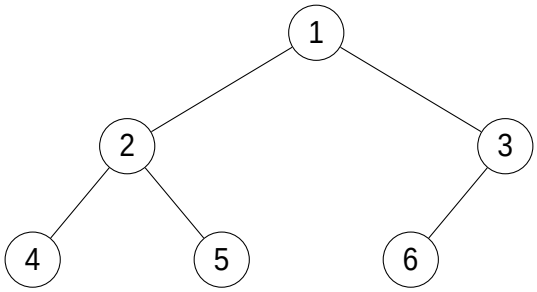
EA4 – Éléments d'algorithmique
TP n° 4

Exercice 1 : arbres binaires représentés par des listes imbriquées

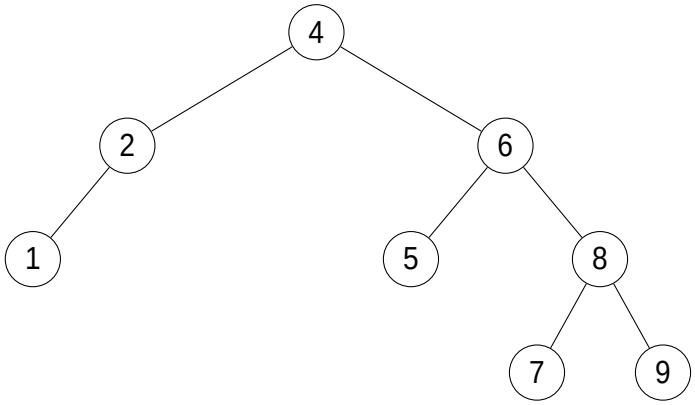
On propose de représenter les arbres binaires par des listes imbriquées telles que

- un arbre vide est représenté par la liste vide `[]`,
- un nœud est représenté par une liste `[Etiquette, FilsGauche, FilsDroit]`,
- une feuille est donc représentée par une liste `[Etiquette, [], []]`.

Ainsi, l'arbre suivant serait représenté par le tableau
`[1,[2,[4,[],[]],[5,[],[]]],[3,[6,[],[]],[]]]`.



1. Écrire le tableau représentant l'arbre suivant :



2. Écrire une fonction qui compte le nombre de nœuds d'un arbre binaire.
3. Écrire une fonction qui calcule la hauteur d'un arbre binaire.
4. Écrire une fonction qui a□che le parcours préfixe d'un arbre binaire.
5. Écrire une fonction qui a□che le parcours infixe d'un arbre binaire.

Exercice 2 : arbres binaires avec des pointeurs

On propose maintenant de représenter les arbres binaires par la donnée, pour chaque nœud, de quatre valeurs : son étiquette et trois pointeurs (identifiants), un vers le fils gauche, un vers le fils droit et un vers le père. Un arbre vide sera représenté par le pointeur `-1`. Par convention la racine aura l'identifiant `0`.
L'arbre en exemple de l'exercice 1 pourra être représenté par les valeurs suivantes que l'on peut considérer être rangées dans un tableau :

```
[[1,  1,  2, -1], # 0
 [2,  3,  4,  0], # 1
 [3,  5, -1, 0], # 2
 [4, -1, -1, 1], # 3
 [5, -1, -1, 1], # 4
 [6, -1, -1, 2]] # 5
```

1. Écrire un tableau représentant l'arbre de la question 1 de l'exercice 1.
2. Écrire un algorithme *non récursif* a□chant les étiquettes des nœuds en ordre préfixe. Il est possible de le faire sans mémoire auxiliaire mais au besoin on pourra s'aider, au choix, d'un tableau de booléens ou d'un tableau de valeurs à 3 états (par exemple 0, 1, 2).

Exercice 3 : opérations supplémentaires et a□chage

À partir d'une des deux représentations proposées, écrire les fonctions suivantes :

1. recherche d'une étiquette ;
2. recherche du maximum ;
3. remplacement d'une étiquette en modifiant l'arbre ;
4. remplacement d'une étiquette en créant un nouvel arbre ;
5. a□chage de gauche à droite, par exemple :

```
      4
   2
      5
1
      6
   3
```

6. a□chage de haut en bas, par exemple :

```
      1
  2       3
4   5   6
```

7. a□chage par une image en utilisant le *package* `graphviz`. Pour cela :
   - générer une sortie de la forme

     ```
     digraph {
       1 -> 2
       2 -> 4
       2 -> 5
       1 -> 3
       3 -> 6
     }
     ```

   - écrire la sortie dans un fichier `arbre.dot` par exemple en redirigeant la sortie du programme par la commande `python3` *fichier*`.py > arbre.dot`,
   - exécuter la commande `dot -T`*format* `arbre.dot >` *fichier_sortie* où *format* peut être `jpg`, `pdf`, `svg`, etc ;
8. conversion d'une représentation version l'autre et vice versa.