❑ **Exercice 1** : *Représentation d'arbres binaires*

1. Dessiner tous les arbres binaires ayant 3 noeuds.
2. Dessiner tous les arbres binaires ayant 4 noeuds.
3. Dessiner un arbre binaire ayant 8 noeuds et de hauteur maximale (resp. minimale).

❑ **Exercice 2** : *Représentation en C*

On rappelle qu'on a défini en C, un arbre binaire (avec des étiquettes entières) par :

```c
#include <stdbool.h>

struct noeud
{
    struct noeud *sag;
    int valeur;
    struct noeud *sad;
};
```

1. Rappeler la définition de la hauteur d'un arbre binaire et écrire une fonction de prototype `int hauteur(ab arbrebinaire)` qui renvoie la hauteur de l'arbre donné en argument.
2. On rappelle que dans cette implémentation, l'espace nécessaire au stockage des noeuds est alloué dynamiquement à l'aide d'instructions `malloc`. Ecrire une fonction de prototype `void libere(ab* arbrebinaire)` qui détruit l'arbre binaire donné en paramètre, en libérant l'espace alloué par ses noeuds. A la fin de l'appel `ab` vaut `NULL`.

❑ **Exercice 3** : *Représentation en OCaml*

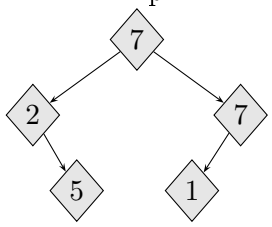
On rappelle qu'on a défini en OCaml un arbre binaire (avec des étiquettes entières) par :

```
type ab = Vide | Noeud of ab * int * ab ;;
```

1. Dessiner l'arbre représenté par :

```
            Noeud(
              Noeud(Noeud(Vide,2,Noeud(Vide,3,Vide)),8,Vide),
              9,
              Noeud(Vide,12,Vide)),
            11,
              Noeud(Noeud(Vide,13,Vide),
              15,
              Vide))
          in
```

2. Donner sa taille et sa hauteur
3. S'agit-il d'un arbre binaire de recherche ? Justifier
4. Donner la représentation en OCaml de l'arbre :



❑ **Exercice 4** : *Un peu de dénombrement*

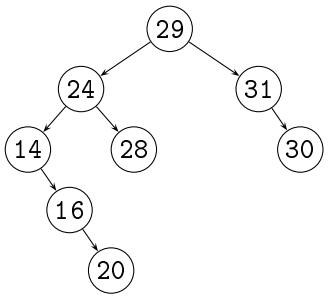
On note T_n le nombre d'arbres binaires à n noeuds.

1. Donner T_0 et déterminer une relation de récurrence liant les $(T_k)_{0\leqslant k\leqslant n}$
   ✪ Utiliser la définition par récurrence des arbres binaires.
2. Vérifier que $T_5 = 42$.
   Le nombre de Catalan d'indice n est défini par :

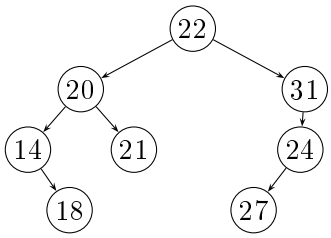
$$C_n = \frac{1}{n+1}\binom{2n}{n}$$

   et on prouve que $T_n = C_n$.

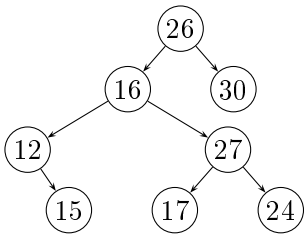
❑ **Exercice 5** : *Parcours d'un arbre binaire*



T_1



T_2



T_3

1. Pour chacun des trois arbres binaires ci-dessus, donner l'ordre des noeuds lors d'un parcours prefixe, infixe et suffixe.
2. Lequel de ces arbres binaires est un ABR ? Justifier

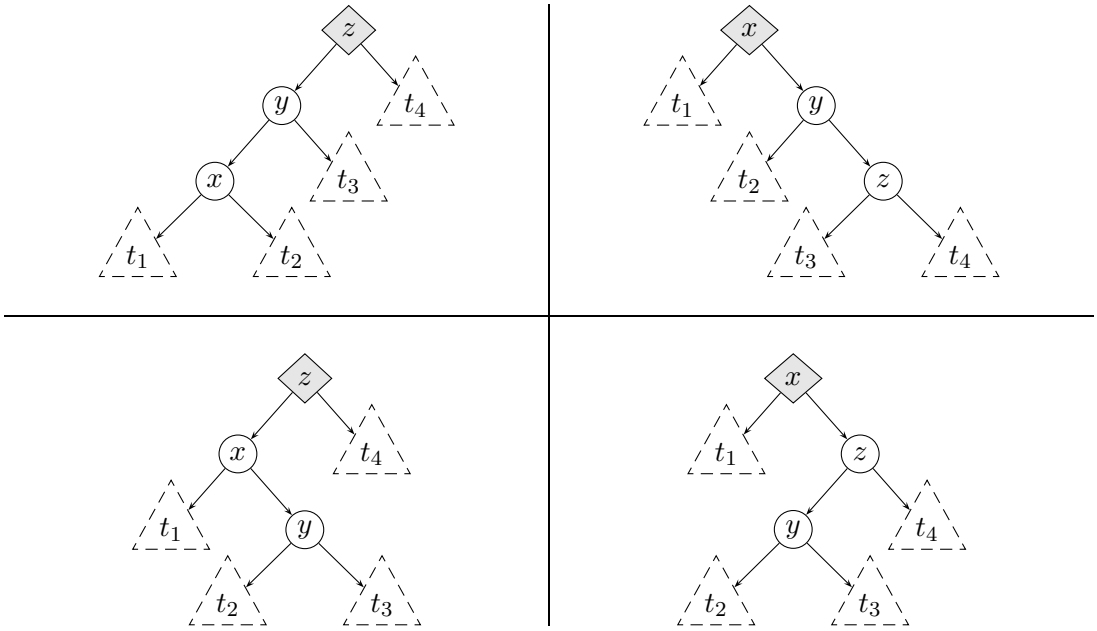
❑ **Exercice 6** : *Un peu de complexité*
On considère la fonction OCaml suivante qui prend en argument un arbre binaire tel que défini par le type de l'exercice 3

```
1  match ab with
2  | Vide -> []
3  | Noeud(g,v,d) ->   v::mystere g @ mystere d;;
```

1. Ecrire une spécification et donner un nom plus approprié à la fonction `mystère`.
2. Rappeler la complexité de l'opérateur `@` et en déduire celle de la fonction `mystere`
3. Proposer une version de cette fonction ayant une complexité linéaire en fonction du nombre de noeuds de l'arbre.
   ✪ Utiliser une fonction auxiliaire avec un accumulateur.

❑ **Exercice 7** : *Equilibrage d'un arbre rouge-noir*

1. Pour insérer un noeud dans un arbre rouge-noir, on commence par utiliser l'algorithme d'insertion usuel dans un ABR et on attribut au nouveau noeud la couleur *rouge*. Quel est alors le seul conflit possible ? (on appellera un tel conflit un *conflit rouge-rouge*).
2. Si le conflit rouge-rouge se situe à la racine, donner une méthode simple pour le résoudre.
3. Si le conflit n'est pas situé à la racine, justifier qu'on se trouve dans l'un des quatres cas suivants où les noeuds rouges sont représentés dans un cercle et les noeuds noirs dans un losange grisé :



4. Montrer qu'en effectuant une ou plusieurs rotations, ces arbres se ramènent à