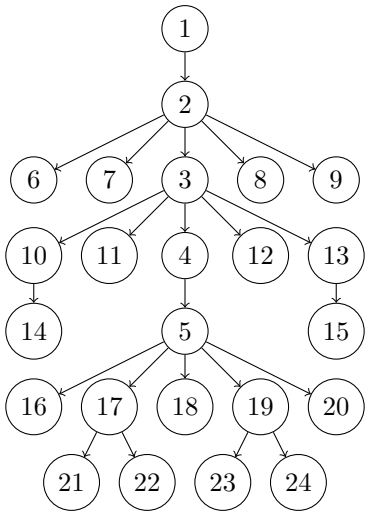
Exercice 1.
On donne l'arbre suivant :



a. Déterminer pour cet arbre, sa racine, sa taille, sahauteur, ses noeuds internes et ses feuilles.

b. Pour le noeud `4`, déterminer son père, ses frères, sa hauteur, sa profondeur.

Exercice 2.
Dessiner tous les arbres binaires ayant respectivement 3 et 4 noeuds.

Exercice 3.
Sachant qu'il y a 1 arbre binaire vide, 1 arbre binaire contenant 1 noeud, 2 arbres bianires contenant 2 noeuds, 5 arbres binaires contenant 3 noeuds et 14 arbres bianires contenant 4 noeuds, calculer le nombre d'arbres binaires contenant 5 noeuds. On ne cherchera pas à les construire tous mais seuleument à les dénombrer.

Exercice 4. ***a.*** Ecrire une fonction `est_feuille(arb)` qui prend en paramètre un arbre binaire et retourne `True` si l'arbre est réduit à une feuille et `False` sinon.

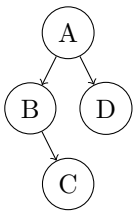
b. Ecrire une fonction `nb_feuilles(arb)` qui prend en paramètre un arbre binaire et retourne le nombre de feuilles de l'arbre.

c. Ecrire une fonction `nb_noeud_int(arb)` qui prend en paramètre un arbre binaire et retourne le nombre de noeuds internes de l'arbre.

d. Ecrire une fonction `parcours_feuilles(arb)` qui prend en paramètre un arbre binaire et affiche les valeurs des feuilles.

Exercice 5.
Ecrire une fonction `appartient(arb,v)` qui retourne `True` si la valeur `v` appartient à l'arbre binaire `arb` et `False` sinon. Quelle est la complexité de cet algorithme ?

Exercice 6. ***a.*** Ecrire une fonction `affiche(arb)` qui imprime un arbre binaire sous la forme suivante : pour un arbre vide, on n'imprime rien ; pour un noeud, on imprime une parenthèse ouvrante, son sous-arbre gauche récurssivement), sa valeur, son sous-arbre droit (récurssivement), puis enfin une parenthèse fermante. Ainsi, pour l'arbre ci-dessous on doit afficher `((B(C))A(D))`.

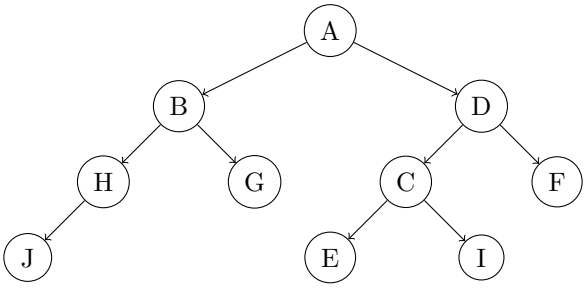


b. Dessiner larbre binaire sur lequel la foncction `affiche` produit la sortie `(1((2)3))`.

Exercice 7.
Ajouter à la classe `Noeud` une méthode `__eq__` permettant de tester l'égalité entre deux arbres binaires à l'aide de l'opération `==`.

Exercice 8.
Ecrire une fonction récursive `parfait(h)` qui reçoit en argument un entier positif `h` et renvoie un arbre binaire parfait de hauteur `h`.

Exercice 9.
On considère l'arbre ci-dessous :



Donner lordre dans lequel les noeuds sont traités dans chacun des cas suivants :

a. Avec un parcours en profondeur d'abord suivant l'odre préfixe.

b. Avec un parcours en profondeur d'abord suivant l'odre infixe.

c. Avec un parcours en profondeur d'abord suivant l'odre postfixe.

d. Avec un parcours en largeur d'abord.

Exercice 10.
Donner cinq arbres de taille 3, différents, dont les noeuds contiennent les valeurs 1, 2 et 3 et pour lesquels un parcours infixe traite les noeuds dans l'odre : `1 2 3`