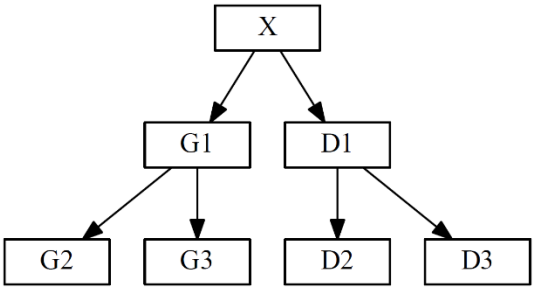
## EXERCICE 4

*Cet exercice porte sur les arbres binaires de recherche.*

Un arbre binaire est soit vide, soit un nœud qui a une valeur et au plus deux fils (le sous-arbre gauche et le sous-arbre droit). Dans la représentation ci-contre,




`X` est un nœud
`G1` est le fils gauche de `X`
`D1` est le fils droit de `X`

Un arbre binaire de recherche est ordonné de la manière suivante :

Pour **chaque** nœud,

- les valeurs de **tous les nœuds** du sous-arbre gauche sont **strictement inférieures** à la valeur du nœud
- les valeurs de **tous les nœuds** du sous-arbre droit sont **supérieures ou égales** à la valeur du nœud

Ainsi, par exemple, toutes les valeurs des nœuds `G1`, `G2` et `G3` sont strictement inférieures à la valeur du nœud `X` et toutes les valeurs des nœuds `D1`, `D2` et `D3` sont supérieures ou égales à la valeur du nœud `X`.

Au fur et à mesure que les billets d'une loterie sont vendus, les numéros inscrits sur les billets sont insérés dans un arbre binaire de recherche. Chaque nœud de l'arbre est un couple, la 1ère valeur correspond au numéro du billet et la seconde est une référence permettant de connaître le lieu de vente du billet.

Par exemple au couple (45,'AZ60') correspond le billet n°45 vendu par le commerçant 'AZ60'. Seul le numéro du billet est utilisé pour positionner le nœud dans l'arbre.

**1.**

**a.** Dessiner l'arbre binaire de recherche dont la racine est (45,'AZ60') et dans lequel on insère dans cet ordre les nœuds (70,'AZ60'), (22,'AZ60'), (65,'BB54'), (58,'BC25') et (67,'BC25')

**b.** Pour construire **la liste triée** de tous les numéros de billets vendus, préciser quel type de parcours de cet arbre faut-il programmer parmi les propositions ci-dessous :

- un parcours en largeur ;
- un parcours en profondeur infixe ;
- un parcours en profondeur préfixe ;
- un parcours en profondeur suffixe.

La fonction `filsgauche(a)` retourne le sous arbre gauche du nœud `a` et `null` si le nœud `a` n'a pas de fils gauche. Il en est de même pour la fonction `filsdroit(a)` avec le fils droit.

**2.** On rappelle que la taille d'un arbre est le nombre de nœuds. Recopier et compléter les ..... de l'algorithme suivant pour que la fonction taille(a) renvoie la taille d'un arbre a.

```
fonction taille(a)
     si a est null
        alors renvoyer .....
     sinon
        renvoyer 1 + ..... + .....
```

**3.** Si `a` n'est pas `null`, la fonction `billet(a)` retourne la première valeur du nœud `a` (c'est-à-dire le numéro du billet) et `reference(a)` retourne la deuxième valeur du nœud `a` (c'est-à-dire le lieu de vente du billet `billet(a)`).

**a.** Que fait la fonction récursive écrite en pseudo-langage suivante, où le paramètre `a` est un nœud et `n` un nombre entier ?

```
fonction mystere(a, n)
  si a est null
    alors renvoyer Faux
  sinon, si billet(a) vaut n
    alors renvoyer Vrai
  sinon
    renvoyer mystere(filsgauche(a)) OU mystere(filsdroit(a))
```

**b.** La fonction précédente est applicable à tout arbre binaire. L'arbre construit à la première question est un arbre binaire de recherche. Recopier et compléter le ..... de la ligne 6 de l'algorithme ci-dessous pour améliorer la fonction `mystere` en tenant compte de cette remarque.

```
1. fonction mystereABR(a, n)
2.      si a est null
3.         alors renvoyer Faux
4.      sinon, si billet(a) vaut n
5.         alors renvoyer Vrai
6.      sinon si .....
7.        renvoyer mystereABR(filsgauche(a))
8.      sinon
9.         renvoyer mystereABR(filsdroit(a))
```