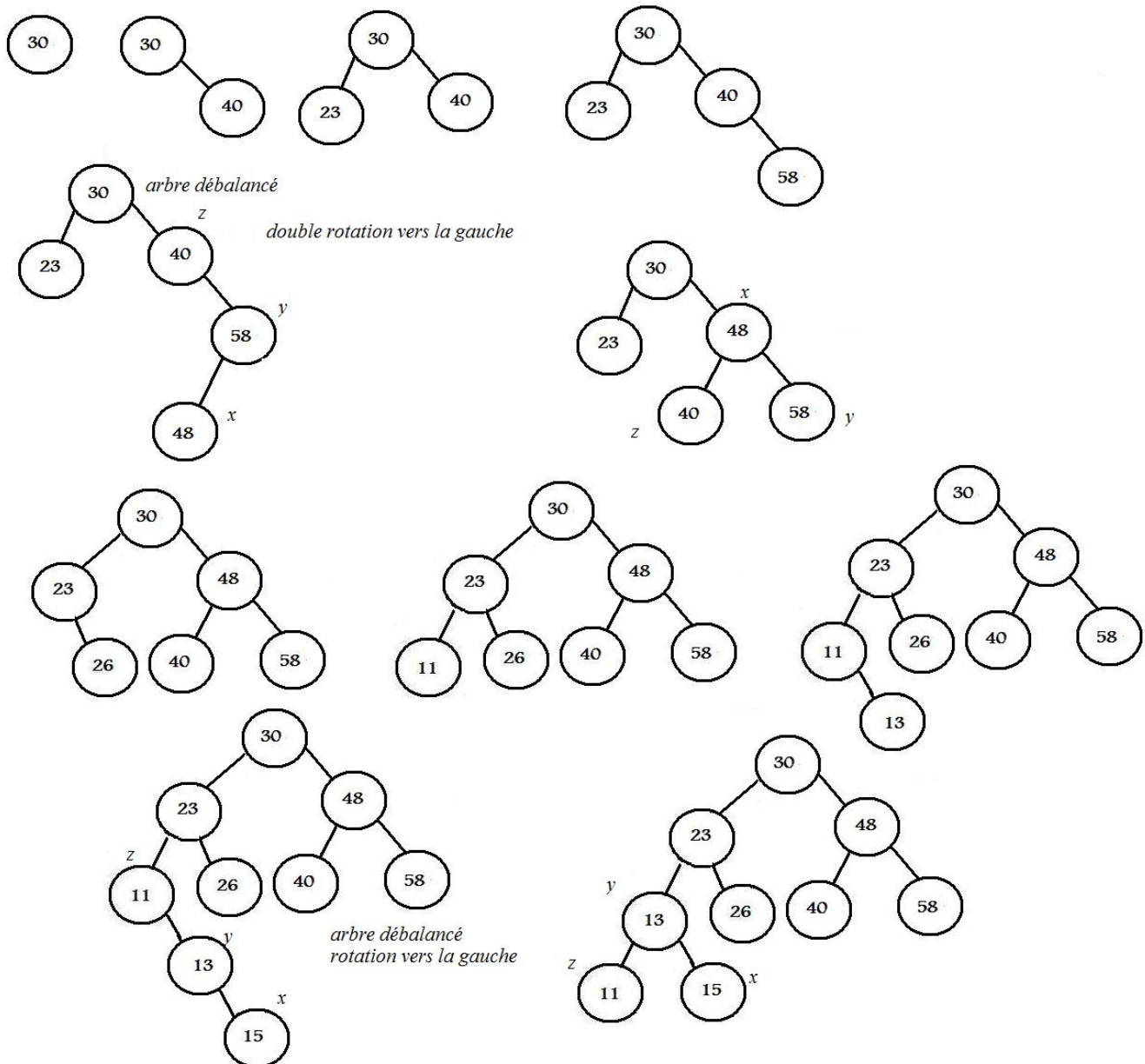
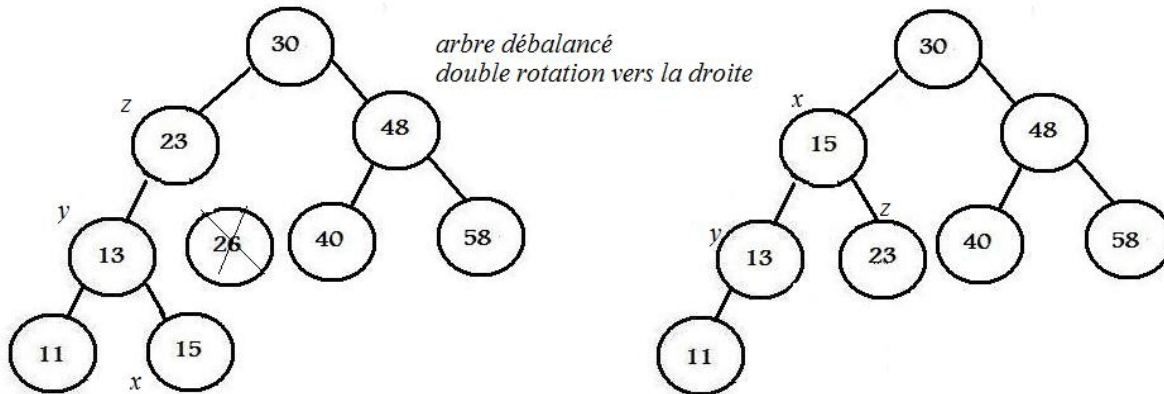


Travaux pratiques: Arbres AVL

1. a) Insérer dans un arbre AVL initialement vide les clés suivantes, dans cet ordre : 30, 40, 23, 58, 48, 26, 11, 13, 15. Dessiner l'arbre après chacune des insertions.



b) Dessiner l'arbre AVL obtenu après la suppression de l'élément de clé 26 dans l'arbre obtenu en a)



2. Étant donné un arbre AVL pour lequel on peut récupérer la hauteur de chaque nœud n avec la méthode `hauteur(n)`, donnez le pseudo-code d'une méthode qui met à jour la hauteur des nœuds situés sur le chemin entre un nouveau nœud inséré (ou un nœud supprimé) et la racine.

On part du nœud inséré et l'on met à jour les hauteurs en remontant.

hauteurAJour(n un nœud)

```

{
  SI ( $n \neq \text{NULL}$ )
  {
    SI ( $n \rightarrow \text{gauche} \neq \text{NULL}$ ) ALORS
      hauteurAJour = max(hauteur( $n$ ), 1+hauteur( $n \rightarrow \text{gauche}$ ));
    SI ( $n \rightarrow \text{droit} \neq \text{NULL}$ ) ALORS
      hauteurAJour = max(hauteur( $n$ ), 1+hauteur( $n \rightarrow \text{droit}$ ));
    hauteurAJour( $n \rightarrow \text{parent}$ );
  }
}

```

3. Étant donné un arbre AVL pour lequel on peut récupérer la hauteur de chaque nœud n avec la méthode $\text{hauteur}(n)$, donnez le pseudo-code d'une méthode qui cherche le premier nœud débalancé (s'il y en a un) entre un nouveau nœud inséré et la racine.

Nous partons la recherche à partir du nœud inséré. Si la différence entre la hauteur du fils droit et celle du fils gauche est plus grande que 1, nous retourner le nœud courant. Sinon, on continue la recherche vers le haut. Nous supposons que la méthode hauteur retourne 0 si le nœud n'a pas d'enfants.

RetourneDebalance(n un nœud)

```
{
    SI ( $n == \text{NULL}$ ) ALORS
        Retourner  $\text{NULL}$ 

    SI ( $n \rightarrow \text{gauche} \neq \text{NULL}$ ) ALORS
         $\text{hgauche} = \text{hauteur}(n \rightarrow \text{gauche})$ 
    SINON
         $\text{hgauche} = 0$ ;
    SI ( $n \rightarrow \text{droit} \neq \text{NULL}$ ) ALORS
         $\text{hdroit} = \text{hauteur}(n \rightarrow \text{droit})$ 
    SINON
         $\text{hdroit} = 0$ ;

     $\text{difference} = \text{hgauche} - \text{hdroit}$ 

    SI ( $\text{difference} \leq 1$ ) ET ( $\text{difference} \geq -1$ ) ALORS
        RetourneDebalance(  $n \rightarrow \text{parent}$ )
    SINON
        Retourner  $n$ 
}
```

4. Soit un arbre AVL implémenté sous forme d'un tableau T, avec la racine en position T[1], et, si un nœud v est en T[i] alors son fils gauche est en T[2i] et son fils droit en T[2i + 1]. Décrivez comment implémenter les méthodes Insérer((k,v),T), Supprimer(k) et Chercher(k) dans ce cas.

Chercher(k): Retourne le nœud ayant une clé k

```
{
    pos = 1;
    TANT QUE (T[pos] → clé != k) ET (T[pos] != NULL)
        SI (k > T[pos] → clé) ALORS
            pos = 2 * pos + 1
        SINON
            pos = 2 * pos
    SI (k == T[pos] → clé) ALORS
        Retourner T[pos]
    SINON
        Retourner NULL
}
```

Insérer((k,v), T)

```
{
    pos = 1;
    TANT QUE (T[pos] != NULL)
        SI (k >= T[pos] → clé) ALORS
            pos = 2 * pos + 1
        SINON
            pos = 2 * pos
    T[pos] = Nouvel Élément (k, v)
    Rebalancer(T)
}
```

Supprimer(k)

```
{
    T[pos] = chercher(k)
    SI T[pos] != NULL ALORS
        Effacer T[Pos] → valeur
        T[pos] = NULL
        Rebalancer(T)
}
```