TD 5
AVL

Question 1 **:** Dans un programme contenant les déclarations :

```
type element=real; (* ou =integer; ou =string[17] *)
     avl=^noeud;
     noeud=record
             fg,fd:avl;
             eq:-1..1;
             cle:element
           end;
```

compléter la procédure

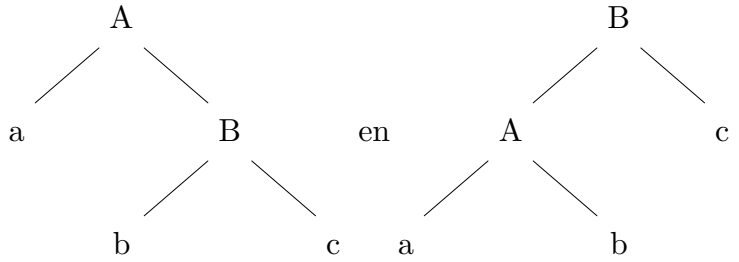
```
procedure verifavl(a:avl);
var x:avl;
function hauteur(a:avl):integer;
begin (* hauteur *)
  ...
end; (* hauteur *)
begin (* verifavl *)
  x:=nil;
  if hauteur(a)=0 then
end; (* verifavl *)
```

La fonction `hauteur(a)` rend la hauteur de l'arbre dont la racine est `a`. Elle calcule d'abord récursivement la hauteur du fils gauche puis celle du fils droit et elle en déduit la hauteur de l'arbre complet. De plus entre le calcul des hauteurs des deux fils, elle copie `a` dans `x`, et juste avant cela elle vérifie que la clé de `a` est bien strictement plus grande que celle de `x` (sauf bien sûr si `x` vaut `nil`). Au moment du calcul de la hauteur, on peut aussi vérifier que `eq` est égal à la différence entre la hauteur du fils droit et celle du fils gauche et qu'il est compris entre -1 et 1.

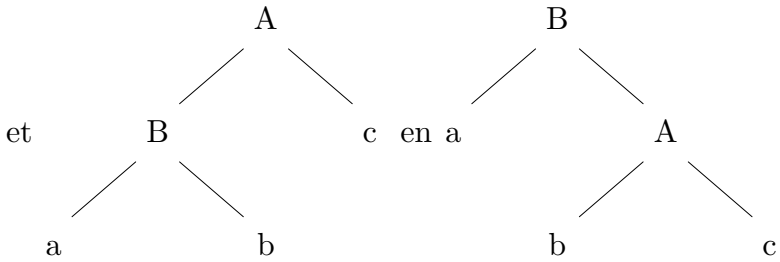
Question 2 **:** Compléter les procédures ayant les entêtes suivants :

```
procedure tourne(var a:avl);
procedure reequilibre(var a:avl);
function ajoute(var a:avl;x:element):boolean;
function enleve(var a:avl;x:element):boolean;
```

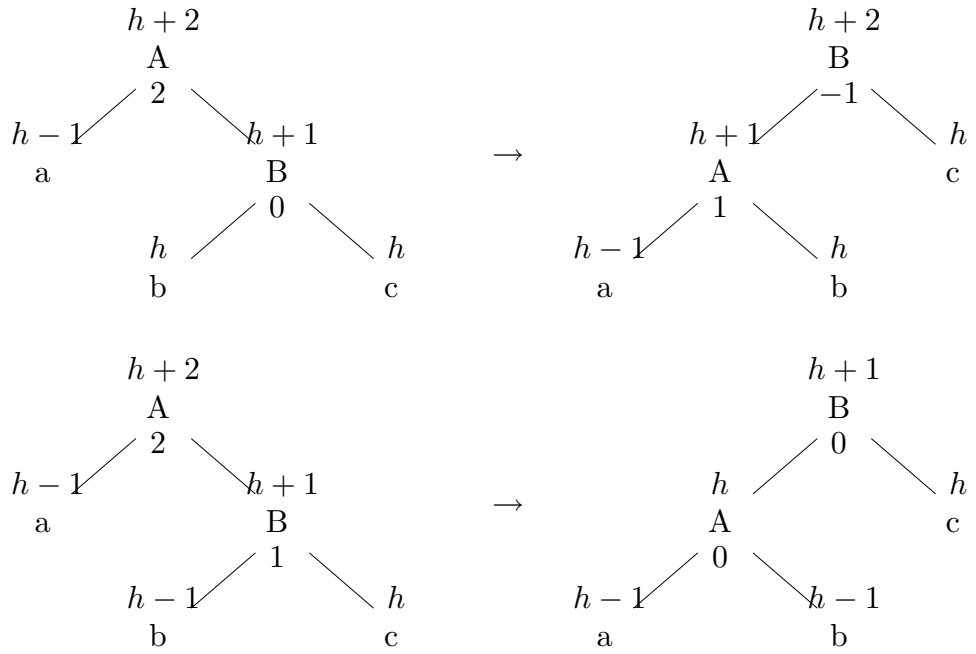
`tourne(A)` transforme l'arbre

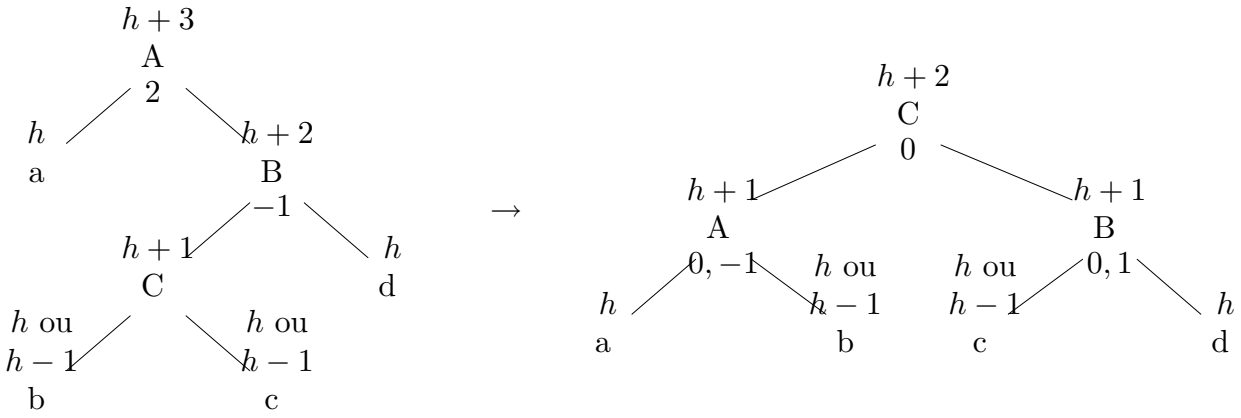


si `A^.eq=1`



sentent des AVL et B un fils de A. Dans les noeuds A et B, seuls les champs fg et fd sont modifiés. L'argument de tourne est modifié : la nouvelle racine de l'arbre est B. reequilibre(A) s'applique à un AVL légèrement déséquilibré car le facteur d'équilibrage de la racine vaut ± 2. Il y a dix rééquilibrages possibles :





La dernière règle compte pour trois, selon les trois valeurs possibles de C^.eq. Les cinq autres règles correspondent à A^.eq=-2. Dessinez les. En fait en entrée A^.eq ne

peut pas valoir ± 2. Donc on supposera que 1 représente 2 et que -1 représente -2. Le rééquilibrage peut toujours être réalisé par un appel à `tourne(A)` ou deux appels `tourne(A^.fd);tourne(A)` suivis d'une mise à jour des champs `eq`. On peut noter que la hauteur globale peut être inchangée ou diminuée d'un, mais qu'elle diminue si et seulement si le facteur d'équilibrage de la racine du nouvel AVL est zéro.

`ajoute(a,x)` insère un nouveau noeud de clé `x` dans l'arbre `a`. Cette expression est vraie si la hauteur de l'arbre augmente d'un, et fausse si la hauteur est inchangée. Cette information est utilisée après un appel récursif si on a inséré `x` dans un des fils de `a`, pour savoir comment mettre à jour `a^.eq` (et si on doit appeler `reequilibre`).

`enleve(a,x)` retire le noeud de clé `x` de l'AVL `a` et rend vrai si la hauteur de l'arbre diminue d'un. Si `x` est la clé de `a` on peut distinguer plusieurs cas selon le nombre de ses fils. Si `a^.fd=nil`, on peut remplacer `a` par son fils gauche. On traite de même le cas où `a^.fg=nil`. Mais si `a` a deux fils, alors il faut copier la plus petite clé du sous-arbre droit de `a` dans `a^.cle` puis l'enlever du sous-arbre droit. Pour trouver le noeud contenant la plus petite clé du sous-arbre droit de `a`, on peut faire `b:=a^.fd;while b^.fg<>nil do b:=b^.fg`.

Question 3 **:** Dessiner la suite des AVL obtenus en partant d'un AVL initialement vide et en y insérant dans cet ordre les éléments 1, 12, 10, 7, 3, 2, 8, 9, 11, 4, 5 et 6. Continuer en enlevant successivement dans cet ordre les éléments 8, 3, 1, 4, 11 et 12. Pour chacun des ajouts et chacune des suppressions, indiquer le nombre de rotations doubles et de rotations simples effectuées.

Question 4 **:** Même question en insérant dans cet ordre dans un AVL initialement vide les éléments 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14 et 15.

Question 5 **:** Ecrire une version itérative d'`ajoute` et d'`enleve`. On utilisera comme pile un tableau de pointeurs dans lequel on mettra tous les noeuds appartenant à la branche de l'AVL allant de la racine jusqu'au noeud ajouté ou enlevé.

Question 6 **:** Lors de l'insertion d'un noeud dans un AVL, soient e_1, e_2, ... e_k les facteurs d'équilibrage des noeuds rencontrés à partir de la racine jusqu'au futur père du noeud à insérer. Soit l le plus grand possible tel que $e_l \neq 0$ ($l = 0$ si les e_i sont tous nuls). Montrer que les facteurs d'équilibrage e_k, e_{k-1}, ... e_{l+1} sont modifiés. Ils passent à ± 1, mais on n'effectue aucun rééquilibrage sur les noeuds correspondants. Montrer que les facteurs d'équilibrage e_1, e_2, ... e_{l-1} sont inchangés et que le seul rééquilibrage éventuel concerne le noeud correspondant à e_l. (Il faut montrer que ce rééquilibrage diminue forcément la hauteur de l'arbre.) Ecrire une version itérative d'`ajoute` qui n'utilise pas de pile. Il suffit pour cela, de garder, lors de la descente dans l'AVL, le dernier noeud rencontré dont le facteur d'équilibrage est non nul, puis de reparcourrir le même chemin dans l'arbre à partir de ce noeud, pour mettre à jour les facteurs d'équilibrage.

Question 7 **:** Compléter la procédure
`procedure desalloue(a:avl);` qui libère la place de tous les noeuds contenus dans `a`.

Question 8 **:** Compléter la procédure

```
const n=1000;
type tab=array[1..n] of element;
procedure triavl(var t:tab);
```

qui trie le tableau `t` en mettant d'abord tous ses éléments dans un AVL.