Travaux Pratiques
séance nr. 5

Arbres binaires de recherche de type AVL

On définit un type "abint" pour un arbre binaire trié d'entiers. On conserve en chaque nœud la hauteur de cet arbre:

```
typedef struct s_abint = {
       T                    etiq;
       unsigned int         h;
       struct s_abint       *ag, *ad;
} *abint ;
```

1. Implanter les opérations : nouveau, enracinement, racine, fils gauche, fils droit, vide, qui ont été donneés en cours.

2. Écrire une fonction « recherche » qui détermine si un élément appartient à un arbre binaire trié. Cette fonction renvoie l'arbre dont la racine est l'élément recherché ou un arbre NUL si l'élément n'est pas trouvé.

3. Écrire une fonction qui détermine si un arbre binaire donné est bien un arbre binaire trié : *est_trie(abint)*. Cette fonction renvoie un booléen. Elle ne vérifie pas l'équilibrage.

4. Écrire une fonction qui affiche les éléments de l'arbre par ordre croissant : affiche(abint)

5. Ecrire les opérations de rotation et double rotation : *rg, rd, rgd, rdg*, sans oublier de mettre à jour les hauteurs stockées dans le champ *h*.

6. Ecrire la fonction qui permet de faire l'insertion d'un élément en préservant l'équilibrage, écrire préalablement la fonction de mesure d'équilibrage *deseq* et la fonction *reeq* qui permet de faire l'équilibrage.

7. Ecrire une fonction qui permet de supprimer un élément de l'arbre. Utiliser et écrire préalablement *otermax1* comme défini en cours. Ne pas oublier de mettre à jour le champ hauteur pour toutes ces fonctions.