Université de Batna 2
Département Informatique
Module : Algorithmique et structures de données

TP 3 : Arbres binaires de recherche

1 Définir une structure **struct** node permettant de coder un nœud d'un arbre binaire contenant une valeur entière. Ajouter des **typedef** pour définir les nouveaux types **node_t** et **tree_t** (ces types devraient permettre de représenter une feuille, c'est à dire un arbre vide).

2 Ecrire une fonction **New_node()** qui prend en argument une valeur entière ainsi que deux arbres et renvoie un arbre dont la racine contient cette valeur et les deux sous-arbres sont ceux donnes en paramètre.

3 Ecrire une fonction (récursive) **Destroy_tree()** qui libère la mémoire occupée par tous les nœuds d'un arbre binaire.

4 Ecrire une fonction (récursive) **Node_number()** qui calcule le nombre de nœuds d'un arbre binaire.

5 Ecrire une fonction **Display_tree ()** qui affiche les valeurs des nœuds d'un ABR par ordre croissant (choisissez le bon type de parcours des nœuds de l'arbre...

6 Ecrire une fonction **Display_tree2 ()** permettant d'afficher les valeurs des nœuds d'un arbre binaire de manière à lire la structure de l'arbre. Un nœud sera affiche ainsi: {g,v,d}ou g est le sous-arbre gauche, v la valeur du nœud et d le sous-arbre droit. indiquent les sous-arbres vides.

7 Ecrire une fonction **Compare()** qui compare deux arbres binaires (la fonction renvoie une valeur nulle si et seulement si les deux arbres binaires ont la même structure d'arbre et qu'ils portent les mêmes valeurs aux nœuds se correspondant).

8 Ecrire une fonction (récursive...) **Insert()** qui ajoute une valeur dans $VABR$ (ce sera un nouveau nœud place correctement dans l'arbre).

9 Ecrire une fonction **Find_node()** qui renvoie l'adresse d'un nœud de $V\ ABR$ donne en paramètre contenant une certaine valeur (ou NULL si cette valeur ne figure pas dans

l'arbre).

10 — Ecrire une fonction **Check_bst** () qui renvoie un entier non nul si et seulement si l'arbre binaire passe en paramètre est un arbre binaire de recherche. *Remarque :* on pourra écrire une fonction auxiliaire (récursive) qui vérifié qu'un arbre binaire (non vide) satisfait les propriétés *d'ABR* et en même temps détermine les valeurs minimales et maximales contenues dans cette arbre binaire (et les renvoie via des pointeurs en argument...).

12—Ecrire une fonction **Delete()** qui supprime une valeur de l'arbre (on supprimera la première rencontrée) tout en conservant les propriétés *d'ABR*. L'algorithme est le suivant (une fois trouve le nœud contenant la valeur en question) :

- si le nœud à enlever ne possède aucun fils, on l'enlève,
- si le nœud à enlever n'a qu'un fils, on le remplace par ce fils,
- si le nœud à enlever a deux fils, on le remplace par le sommet de plus petite valeur dans le sous-arbre droit, puis on supprime ce sommet.