TD 05 : Arbres Binaires de Recherche

Consignes générales : N'oubliez pas pour ce TD comme pour les suivants de vous créer un répertoire consacré au TD et d'enregistrer vos codes dedans.

On rappelle que les commandes à taper dans le terminal pour compiler puis éxécuter votre programme C :
— Pour compiler : *gcc -o nom_executable nom_programme.c*
— Pour exécuter : *./nom_executable*

Exercice 1 (*Question de cours*)

1. Construire un arbre binaire en insérant les éléments dans cet ordre ; 10, 3, 5, 15, 20, 12, 7, 45, 9.
2. Construire un second arbre binaire en insérant les éléments dans l'ordre inverse que précédement. Obtient-on le même arbre ?
3. Effectuer un parcours infixe sur ces deux arbres. Que remarque-t-on ?
4. Supprimer l'élément 5 puis 12 du premier arbre.

Exercice 2 (*Construction d'un ABR*)
Les versions pseudo-code des fonctions demandées aux question 2,3 et 5 sont dans le cours.

1. Définir la structure permettant de construire un arbre binaire contenant des entiers.
2. Écrire la fonction `recherche(pArbre a, int e)` indiquant si l'élément `e` appartient à l'ABR pointé par `a`.
3. Écrire la fonction **recursive** `insertABR(parbre a, int e)` permettant d'inserer l'élément `e` dans l'ABR. Attention, l'insertion doit respecter les règles des ABR !
4. Écrire une version **itérative** de la fonction d'insertion écrite précédement.
5. Écrire les fonctions nécessaires à la suppression d'un élément dans un ABR.
6. Creer un ABR et insérer les éléments suivants dans cet ordre : 10, 3, 5, 15, 20, 12, 7, 45, 9. Afficher l'arbre et vérifier qu'il s'agit bien d'un ABR.
7. Vérifier si les élément 13 et 12 appartiennent à l'arbre.
8. Supprimer l'élement 15 et vérifier que l'arbre et toujours un ABR.

Exercice 3 (*Test des fonctions de recherche*) Pour cet exercice, on travaille sur l'arbre construit dans l'exercice précédent.

1. Modifier la fonction de recherche `recherche(pArbre a, int e)` pour qu'elle retourne le nombre de nœuds qui auront été parcourus lors de cette recherche. Si un élément n'appartient pas à l'arbre, on affichera que l'élément recherché n'existe pas mais on retournera tout de même le nombre de nœuds visités.
2. Proposer une modification de la fonction permettant le parcours Préfixe de l'arbre pour que cette dernière serve à rechercher si un élément existe dans l'arbre en parcourant l'arbre et en s'arrêtant lorsque l'élément est trouvé. Comme pour la question précédente, la fonction doit afficher le nombre de nœuds parcourus.
3. Aficher le nombre de nœuds parcourus pour les deux fonctions lors de la recherche des éléments suivants : 10, 20, 22.

Exercice 4 (*ABR ?*)
Proposer une fonction permettant de vérifier si un arbre binaire est un ABR ou non.

Exercice 5 (*D'arbre binaire à ABR*) Proposer une ou plusieurs fonctions qui permettent de construire un ABR à partir d'un arbre binaire simple.

Exercice 6 (*Tri de tableau*)

1. Déclarer un tableau de taille 15 et le remplir avec des valeurs saisies *différentes*.
2. Construire un ABR à partir de ce tableau
3. Trier ce tableau en vous basant sur un parcours de l'ABR.
4. *bonus* : reprendre la question 1 mais les valeurs du tableau sont aléatoires entre 0 et 100 et ne doivent pas avoir de doublons !