TD 3 : arbres binaires

Exercice 1. Parcours en largeur

Écrire une fonction permettant de faire un parcours en largeur d'un sous-arbre a.

Exercice 2. Compter les feuilles

Écrire une fonction **int compter_feuilles(t_abre a)** qui renvoie le nombre de feuilles d'un arbre binaire a.

Exercice 3. Hauteur

Donner une définition récursive de la hauteur d'un arbre ***a***. En utilisant cette définition, écrire une fonction entière récursive **hauteur** qui délivre la hauteur d'un sous-arbre binaire (par convention, la hauteur d'un arbre vide vaut 0).

Exercice 4. Profondeur

a) Écrire une fonction entière *itérative* **profondeur** qui délivre la profondeur d'un nœud quelconque.
b) Écrire la version récursive
c) Écrire une fonction **affiche_prof_feuille** qui affiche la valeur et la profondeur de chacune des feuilles de l'arbre.

Exercice 5. Somme et Moyenne

Écrire une fonction calculant la somme des valeurs d'un arbre.
Écrire une fonction calculant la moyenne des valeurs d'un arbre.

Exercice 6. Contient et cherche

Écrire une fonction booléenne **contient** qui prend en paramètre un entier et vaut vrai si cette valeur est présente dans le sous-arbre ***a***, faux sinon.
Quelles modifications faut-il apporter à cette fonction pour la transformer en fonction **cherche** qui renvoie la racine de l'arbre trouvé ?

Exercice 7. Arbre Binaire de Recherche

Écrire une fonction qui vérifie qu'un sous-arbre ***a*** est bien un arbre binaire de recherche.

Exercice 8. ABR équilibré et de hauteur maximale

On considère tous les nombres compris entre 1 et 100. Dans quel ordre faut-il insérer ces nombres dans un ABR afin d'obtenir :

a) un arbre complètement déséquilibré, c'est-à-dire de hauteur maximale
b) un arbre équilibré, c'est-à-dire le moins haut possible.

Exercice 9. Max d'ABR

Écrire une fonction itérative qui renvoie le plus grand élément d'un arbre binaire de recherche donné. Donner une version récursive.