TD 2 : arbres binaires

Préambule : nous allons utiliser la structure suivante pour représenter un arbre

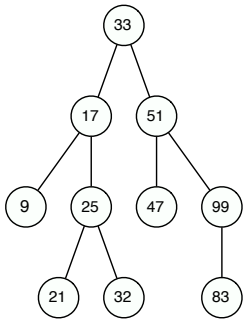
typedef struct nœud { int val ; struct nœud*pere, *sag, *sad ;}
typedef struct nœud* t_arbre ;

Exercice 1

Écrire les primitives d'accès pour une mise en œuvre par pointeur.

Rappel des primitives :
creer_arbre, supprimer_arbre, arbre_vide, pere, sag, sad, est_feuille, est_racine, val_racine, modif_racine, ajout_gauche, ajout_droit
Travail à la maison : finir les implémentations.

Exercice 2



a) Donner les énumérations des valeurs de l'arbre binaire correspondant respectivement à un parcours préfixe, postfixe et infixe

b) Écrire trois fonctions **parcours_prefixe**, **parcours_postfixe** et **parcours_infixe** qui effectuent respectivement les parcours canoniques préfixe, postfixe et infixe d'un arbre binaire, en appliquant une fonction de prototype **void (*)(int*)** à chaque noeud.

c) Ecrire 1 programme principal permettant d'afficher un arbre (initialisé rapidement), puis de doubler les valeurs de chaque nœud, puis de réafficher l'arbre. Vérifiez l'implémentation.

Exercice 3

On considère un arbre binaire représentant une expression arithmétique syntaxiquement correcte : les feuilles sont des entiers, les nœuds non feuilles sont des caractères représentant des opérateurs ('+', '-', '*' ou '/'), et tout nœud non feuille possède exactement deux fils.
Les nœuds de l'arbre contiennent des valeurs utiles de type union défini par :

typedef union { int ande ; char ateur ; } t_oper ;

Écrire une fonction entière **eval** qui calcule et délivre la valeur de l'expression arithmétique.