Série 1

Arbres binaires de recherche

Exercice 1

Définir une structure struct_noeud_s permettant de coder un noeud d'un arbre binaire contenant une valeur entière.

typedef struct noeud_s {

int valeur ;

struct noeud_s ∗gauche ;

struct noeud_s ∗ d r o i t ;

};

typedef struct noeud_s ∗ nœud,

Exercice 2

Écrire une fonction cree_arbre() 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 donnés en paramètre.

#include <stdlib.h>

nœud creer_arbre (int val eur , noeud gauche , noeud d r o i t) {

noeud arbre = malloc (sizeof (struct noeud_s)) ;

arbre−>v al eur = valeur ;

arbre−>gauche = gauche ;

arbre−>droit = droit ;

return arbre ;

}

Exercice 3

Écrire une fonction (récursive) detruit_arbre() qui libère la mémoire occupée par tous les noeuds d'un arbre binaire.

```
#include <s t d l i b . h>

void detruit_arbre ( nœud cellule ) {

if ( cellule  == NULL)

return ;

detruit_arbre ( cellule−>gauche ) ;

d etruit_arbre ( cellule−>d r o i t ) ;

free ( cellule) ;

}
```

Exercice 4

Écrire une fonction (récursive) nombre_de_noeuds() qui calcule le nombre de noeuds d'un arbre binaire.

Exercice 5

Écrire une fonction affiche_arbre() qui affiche les valeurs des noeuds d'un ABR par ordre croissant (choisissez le bon type de parcours des noeuds de l'arbre. . .).

Exercice 8

Écrire une fonction (récursive. .) insere () qui ajoute une valeur dans l'ABR (ce sera un nouveau noeud placé correctement dans l'arbre).

Exercice 9

Écrire une fonction trouve_noeud() qui renvoie l'adresse d'un noeud de l'ABR donné en paramètre contenant une certaine valeur (ou NULL si cette valeur ne figure pas dans l'arbre).