Examen – Session de rattrapage

Matière : Atelier programmation II Enseignants : Majdi JRIBI et Dorsaf SEBAI Filière : MPI Nombre de pages : 03 pages	Semestre: Second semestre Date: 29 Juillet 2019 Durée: 1h30 Documents : ***non autorisés***

Chaque exercice doit être obligatoirement rédigé sur une feuille d'examen à part

Exercice 1 : Arbre binaire

Considérons les déclarations suivantes d'un arbre binaire:

```
typedef struct arb
    {       int val;
            struct arb *fg;
            struct arb *fd;
     } arb;

typedef arb* arbre;
```

1- Ecrire en langage C une fonction récursive ***void detruire_arbre (arbre ar)*** qui prend en entrée un arbre binaire et qui permet de libérer la mémoire occupée par tous les nœuds de cet arbre.

2- Ecrire en langage C une fonction **itérative** qui cherche un élément dans un arbre binaire **quelconque.** Cette fonction retourne 1 si l'élément est trouvé et 0 sinon.

Exercice 2 : les files

Considérons les déclarations suivantes d'une file d'entiers:

```
typedef struct Noeud
{
int valeur;
struct Noeud * suivant;
} Noeud;

Typedef Noeud * file;
```

Soient les fonctions prédéfinies suivantes concernant des traitements sur les files:

- *Tester si une file est vide* : ***int estvide(file fil);*** (Elle permet de envoyer 0 si la file est vide).
- *Enfiler un élément dans une file* : ***file enfiler(file fil, int val);*** (Elle permet d'ajouter un entier à la queue de file).
- *Défiler un élément d'une file* : ***file defiler(file fil);*** (Elle permet de supprimer la tête de la file et rend la nouvelle file sans la tête).
- *Déterminer la tête de la file:* ***int tete(file fil);*** (Elle récupère la valeur de la tête de la file).

Soient $\mathbf{F_1}$ et $\mathbf{F_2}$ deux files d'entiers. Chaque file est triée par ordre croissant (l'élément le plus petit se trouve à la tête de la file).

En utilisant seulement les fonctions prédéfinies citées, écrire une fonction en langage C ***file fusion (file F_1, file F_2)*** qui permet de fusionner les deux files $\mathbf{F_1}$ et $\mathbf{F_2}$ dans la file $\mathbf{F_2}$ de telle sorte que la file finale soit triée par ordre croissant (la tête de la file finale contient l'élément le plus petit).

Exercice 3 : Arbre binaire

Considérons les déclarations suivantes d'un arbre binaire:

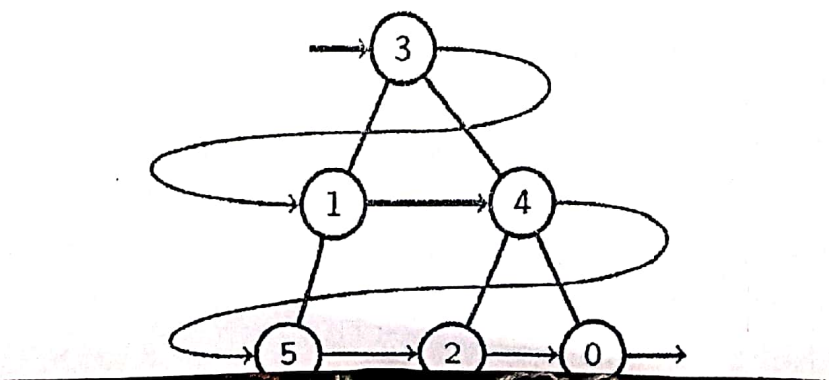
```
typedef struct arb
        {       int val;
                struct arb *fg;
                struct arb *fd;
        } arb;

typedef arb* arbre;
```

Il existe deux types de parcours pour un arbre binaire : Un parcours en profondeur et un parcours en largeur. Nous avons déjà vu en cours le parcours en profondeur qui lui-même se décompose en trois sous parcours : préfixé, infixé et postfixé. Le parcours en largeur est réalisé quant à lui de la manière suivante : on commence par explorer la racine, puis ses fils, puis les fils des fils, etc.

Remarque : l'exploration des fils d'un nœud se fait de gauche à droite

Exemple de parcours en largeur d'un arbre binaire :



Si lors de ce parcours en largeur on affiche les valeurs des nœuds, le résultat obtenu est le suivant :

3, 1, 4, 5, 2, 0

Ecrire une fonction **itérative** en langage C qui permet de réaliser le parcours en largeur d'un arbre binaire et afficher les valeurs de ses nœuds suite à ce parcours.