

Examen – Session principale

Matière : Atelier programmation II
Enseignants : Majdi JRIBI et Dorsaf SEBAI
Filière : MPI
Nombre de pages : 6 pages

Semestre: Second semestre
Date: 17 Mai 2017
Durée: 1h30
Documents : non autorisés

Les réponses doivent être rédigées obligatoirement sur les feuilles de réponse (pages 5 et 6)

L'examen contient 6 pages. Seulement les pages 5 et 6 sont à rendre.

Exercice 1 : Liste chainée

Considérons les déclarations suivantes :

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

```
Typedef Noeud * liste;
```

Déterminer le rôle de chacune des fonctions suivantes (Répondre sur les feuilles de réponses)

Question 1

```
// Ici les listes lst1 et lst2 sont triées par ordre
croissant
void fonction_1 ( liste lst1 , liste lst2 , liste *
p_lst ) {
    liste lst ;
    if ( lst1 == NULL )
        * p_lst = lst2 ;
    else if ( lst2 == NULL )
        * p_lst = lst1 ;
    else
    {
        if ( lst1 -> valeur < lst2 -> valeur )
        {
            fonction_1 ( lst1 -> suivant , lst2 , & lst ) ;
            lst1 -> suivant = lst ;
            * p_lst = lst1 ;
        }
        else
            fonction_1 ( lst1 , lst2 -> suivant , & lst ) ;
            lst2 -> suivant = lst ;
            * p_lst = lst2 ;
    }
}
```

Question 2

```
void fonction_2 ( liste lst , liste * p_lst1 , liste
* p_lst2 )
{
    if ( lst == NULL || lst -> suivant ==
NULL )
    {
        * p_lst1 = lst ;
        * p_lst2 = NULL ;
    }
    else
    {
        liste lst1 = NULL , lst2 = NULL ;
        fonction_2 ( lst -> suivant -> suivant ,
& lst1 , & lst2 ) ;
        lst -> suivant -> suivant = lst2 ;
        * p_lst2 = lst -> suivant ;
        lst -> suivant = lst1 ;
        * p_lst1 = lst ;
    }
}
```

Question 3

```
void fonction_3 ( liste * p_lst )
{
    liste lst1 , lst2 ;
if ((* p_lst) != NULL && (* p_lst) → suivant != NULL )
{
    fonction_2 (* p_lst , & lst1 , & lst2 );
    fonction_3 (& lst1 );
    fonction_3 (& lst2 );
    fonction_1 ( lst1 , lst2 , p_lst );
}
}
```

Question 4

```
liste fonction_4 (liste * pprem, int a)
{ liste x;
if ( *pprem == NULL) return(NULL);
else
if ( (*pprem) → valeur != a )
return(fonction_4 ( & (*pprem) → suivant
),a);
else
{ x = * pprem;
* pprem = (*pprem) → suivant;
return ( x );
}
}
```

Question 5

```
liste fonction_5(liste l, int n) {
    liste R;
    if (l == NULL) {

        return (l);
    }
    if (l → valeur == n) {
        R = l;
        l = l → suivant;
        free(R);
        return (l);
    }
    else {
        l → suivant = fonction_5 (l → suivant,n);
        return (l);
    }
}
```

Question 6

```
int fonction_6(liste l, int val)
{
    liste p=l;
    if (p == NULL) {
        return (0);
    }
    else if (p → valeur == val) {
        return (1);
    }
    else {
        return (fonction_6(p → suivant,
val));
    }
}
```

Question 7

```
void fonction_7(liste la)
{
    liste p,q,h;
    int nb;
    p=la;
    while(p → suivant!=NULL){
        q=p;
        h=p → suivant;
        nb=0;
        while(h!=NULL){
            if(h → val == p → val){
                if(nb<2){
                    nb++;
                    h=h → suivant;
                    q=q → suivant;
                }
            }
            else{
                q → suivant=h → suivant;
                free(h);
                h=q → suivant;
            }
        }
        else{
            h=h → suivant;
            q=q → suivant;
        }
    }
    p=p → suivant;
}
```

Exercice 2 : Arbre binaire et arbre binaire de recherche

Considérons le code écrit en langage C qui concerne des traitements sur les Arbre binaire et arbre binaire de recherche.. Répondre aux questions qui figurent sur les feuilles de réponses

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

```
typedef arb* arbre;
```

Question 1

// Ici A est un arbre binaire

```
arbre fonction_1(arbre A)
{
    int tmp;
    arbre y;
    if(A!=NULL && (A->fg)!=NULL)
    {
        y = A->fg;
        tmp = A->val;
        A->val = y->val;
        y->val = tmp;
        A->fg = y->fg;
        y->fg = y->fd;
        y->fd = A->fd;
        A->fd = y;
    }
    return(A);
}
```

Question 3

// Ici les arbres ar1 et ar2 sont des arbres binaires

```
int fonction_3 ( arbre ar1 , arbre ar2 )
{
    if( ar1 == NULL)
        return ( ar2 != NULL );
    else
    {
        if( ar2 == NULL)
            return 1 ;
        else
            return (( ar1->val != ar2->val )
|| fonction_3 ( ar1->fg , ar2->fg )
|| fonction_3 ( ar1->fd , ar2->fd ) );
    }
}
```

Question 2

// Ici ar est un arbre binaire

```
void fonction_2 ( arbre ar )
{
    if( ar == NULL)
        printf( "_" );
    else {
        printf( "{" );
        fonction_2 ( ar->fg );
        printf( ",%d , " , ar->val );
        fonction_2 ( ar->fd );
        printf( "}" );
    }
}
```

Question 4

// Ici l'arbre ar est binaire de recherche

```
arbre fonction_4 ( arbre ar , int va ) {  
    arbre noeud = ar , * pere = &ar ;  
    arbre nouveau_noeud , *nouveau_pere ;  
    while ( noeud != NULL) {  
        if ( va == noeud->val)  
            break ;  
        if ( va < noeud->val) {  
            pere = &noeud->fg ;  
            noeud = noeud->fg ;  
        } else {  
            pere = &noeud->fd ;  
            noeud = noeud->fd ;  
        }  
        if ( noeud != NULL) {  
            if ( noeud->fg == NULL) {  
                if ( noeud->fd == NULL) {  
                    *pere = NULL;  
                    free ( noeud ) ;  
                } else {  
                    * pere = noeud->fd ;  
                    free ( noeud ) ;  
                }  
            } else {  
                if ( noeud->fd == NULL) {  
                    * pere = noeud->fg ;  
                    free ( noeud ) ;  
                } else {  
                    nouveau_noeud = noeud->fd ;  
                    nouveau_pere = &noeud->fd ;  
                    while ( nouveau_noeud != NULL)  
                        if ( nouveau_noeud->fg != NULL) {  
                            nouveau_pere = &nouveau_noeud->fg ;  
                            nouveau_noeud = nouveau_noeud->fg ;  
                        }  
                        noeud->val = nouveau_noeud->val ;  
                        *nouveau_pere = nouveau_noeud->fd ;  
                        free ( nouveau_noeud ) ;  
                }  
            }  
        }  
    }  
    return ar ;  
}
```

Examen – Session principale

Matière : Atelier programmation II
Filière : MPI
Nom :
CIN :

Semestre: Second semestre
Date: 17 Mai 2017
Prénom :
Numéro Inscription :

Feuilles de réponse

Exercice 1 :

Donner le rôle de chacune des fonctions :

- Question 1 : fonction _1

.....
.....
.....

- Question 2 : fonction _2

.....
.....
.....

- Question 3 : fonction _3

.....
.....
.....

- Question 4 : fonction _4

.....
.....
.....

- Question 5 : fonction _5

.....
.....
.....

- Question 6 : fonction _6

.....
.....
.....

- Question 7 : fonction _7

.....
.....
.....

Ne rien écrire ici

Exercice 2 :

1- Donner le rôle de chacune des fonctions :

- Question_1 : fonction_1

- Question_2 : fonction_2

- Question_3 : fonction_3

- Question_4 : fonction_4

Matière
Enseigné
Filière
N°