المعهد الوطني للعلوم التطبيقية والتكنولوجيا
Insat

Examen – Session principale

Matière : Atelier programmation II Enseignants : Majdi JRIBI et Dorsaf SEBAI Filière : MPI Nombre de pages : 7 pages	Semestre: Second semestre Date: 16 Mai 2019 Durée: 1h30 Documents : ***non autorisés***

Les réponses doivent être rédigées obligatoirement sur les feuilles de réponse (pages 6 et 7)

L'examen contient 7 pages. Seulement les pages 6 et 7 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 )
                {
        fontion_2  (* p_lst , & lst1 , & lst2 ) ;
        fonction_3 (& lst1 ) ;
        fonction_3 (& lst2 ) ;
        fonction_1 ( lst1 , lst2 , p_lst ) ;
                }
}
```

Question 4

```
liste  fonction_4(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_4 (l→suivant,n);
      return  (l);
   }
}
```

Question 5

```
liste fonction_5 (liste * pprem, int a)
{ liste x;
if ( *pprem == NULL) return(NULL);
else
if ( (*pprem) → valeur != a )
return(fonction_5 (  & (*pprem) → suivant
),a);
else
{ x = * pprem;
* pprem = (*pprem) →suivant;
return ( x );
}
}
```

Question 6

```
void fonction_6(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 : Liste chainée

Considérons le code écrit en langage C qui concerne des traitements sur les listes chainées.
Donner le rôle de chaque focntion

```
#include <stdio.h>
#include <stdlib.h>

typedef struct element * Pelement;

typedef struct liste * FListe;

typedef struct element
{
int x;
 Pelement suivant;
}Element;

typedef struct liste
{
 Pelement premier;
 Pelement courant;
 Pelement dernier;
}Liste;
```

Question 1

```
void Traitement_1(FListe L)
{
 L = (FListe) malloc ( sizeof(Liste));
 L→premier = (Pelement) malloc (
sizeof(Element));
 L→courant = (Pelement) malloc (
sizeof(Element));
 L→dernier = (Pelement) malloc (
sizeof(Element));
 L→premier = NULL;
 L→courant = NULL;
 L→dernier = NULL;
}
```

Question 2

```
void Traitement_2(FListe L, Pelement
nouveau)
{
 Nouveau→suivant = L→premier;
 L→premier = nouveau;
if( L→dernier ==NULL )
 L→dernier = nouveau;
}
```

Question 3

```
void Traitement_3(FListe L, int n)
{
 printf("***** Traitement_3 *****\n");
int i;
 Pelement Pel;
for(i=1; i<=n; i++){
 Pel = (Pelement)malloc( sizeof
(Element));
 Pel→x = i;
 Traitement_2(L, Pel);
 }
}
```

Exercice 3 : **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.** Donner le rôle de chaque fonction.

```
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 2

```
// Ici l'arbre ar est un arbre binaire et
ar !=NULL

int fonction_2 ( arbre ar , int *min , int
*max)
{
int i ;
*min = *max = ar→val ;
if ( ar→fg != NULL)
if ( !fonction_2 ( ar→fg , &i , max) || ! (
    ar→val > *max) )
return 0 ;
if ( ar→fd != NULL)
if ( ! fonction_2 ( ar→fd , min , &i ) || ! (
ar→val <= *min ) )
return 0 ;
return 1 ;
}
```

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 ) ) ;
}
}
```