Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
UNIVERSITE M'HAMED BOUGARA –BOUMERDES
FACULTE DES SCIENCES
DEPARTEMENT D'INFORMATIQUE

Nature de l'examen : ETLD
Module : ASD

Année : 2016-2017
D. Hadjidj, L. Iza, D. Salhi

Exercice 1 (4 pts)

Ecrire une fonction permettant de créer à partir d'un tableau dynamique d'entiers A[n][n] un tableau de listes linéaires simples tab[n] où tab[i] , i=0,..n-1, représente la tête de la liste contenant les éléments de la ligne i de A.

Exercice 2 (a-3 pts, b- 7 pts)

Soit un tableau B[N] de listes linéaires simples, contenant la description de processus (programmes) dans un système multi-utilisateurs. Chaque processus étant décrit par une structure comportant un **id** (entier), et une **priorité** (entier de 1à m).

a- Ecrire une fonction permettant de vérifier si une liste de tête B[i] contient des éléments de même priorité.

b- Ecrire une fonction permettant, en une seule passe, de supprimer de la liste de tête B[i] le processus ayant un **id** égal à 15 et une **priorité** égale à m, s'il est présent.

Exercice 3 (a- 3 pts, b- 3 pts)

Soit un arbre binaire de tri non vide dont le pointeur vers la racine est R et où les clés sont des entiers.

a. Écrire une fonction permettant de remplacer l'élément de clé maximale de cet arbre par un élément de clé inferieure à sa clé minimale.

b. Ecrire une fonction itérative pour afficher les sommets feuilles de l'arbre.

NB : Donner les types des structures utilisées et les appels des fonctions.

UMBB-Faculté des Sciences
Département d'Informatique

Année 2016-2017
Module : ASD

Barème détaillé

Exercice 1 (4 pts)

Déclarations + entête de la fonction +appel ----------1.5pts
Réservation du tableau +remplissage ----------2.5pts

Exercice 2(3 pts+7 pts)

a) Déclarations + entête de la fonction +appel ------1.5 pts
   Vérification de la liste ----------1.5 pts

b) Appel +entête de la fonction ---------- 1 pts
   Suppression ----------6 pts

Exercice 3 (3 pts+3 pts)

a) Déclarations + entête de la fonction +appel -------1.5 pts
   Remplacement----------1.5 pts
b) Entête de la fonction + appel----------1 pt
   Affichage ---------- ----------2 pts

Corrigé de l'ETLD

Exercice 1

Déclarations :

```
const int n=… ;
struct element
{
     int info;
     element* suivant;
};
-
-
-
element* tab[n];
int** A ;
```

Appel à la fonction :

```
convert(A, tab) ;
```

Fonction :

```
void convert(int** A ,element* tab[])
{
for(int i=0 ; i<n ;i++)
{tab[i]=0;
 for( int j=n-1;j>=0;j--)
 insertDebut(tab[i],A[i][j]);
}
}
```

Exercice 2

a)

Déclarations :

```
struct processus
{
     int id;
     int priorite;
     processus*suivant ;
};
-
-
-
```

Appel à fonction :

```
bool y= samePriority(B[i]) ;
```

Fonction :

```
bool samePriority (processus *L)
{
Processus *courant1= L, *courant2;
bool same = false ;
while((courant1 != 0) &&(!same))
{courant2=courant1->suivant;
 while (( courant2 !=0) && (!same))
     if(courant1 ->priorite == courant2->priorite)
       same= true ;
     else
      courant2= courant2->suivant;
 if((courant1 != 0) &&(!same)) courant1=courant1->suivant;
}
return same ;
}
```

b)

Appel à la fonction :

```
supProcessus(B[i],15,m);
```

Fonction :

```
supProcessus(processus* &debut,int iden,int prio);
{ processus* courant= debut;
while (courant !=0)
   if ((courant->id!=iden)||(courant->priorite!=prio))
        courant=courant->suivant;
     else
   {
     if (courant==debut){supDebut(debut);courant=debut;}
      else
       if( courant->suivant==0){supFin(debut);courant=0;}
          else
            supMilieux(courant);
   }
}
```

Exercice 3 :

a)

Déclarations :

```
     struct sommet
     {    int cle;
          arbre* petit,*grand ;
     };

arbre* R ;
```

Appel à la fonction :

```
modif(R,M) ;
```

Fonction :

```
void modif(arbre* &R, int M)
{if(R !=0)
{
  sommet* l=R ;
  while (l->petit!0)l=l->petit;
  l->petit= new sommet;
  l->cle= M;l->petit=0;l->grand=0;

  if (R->grand !=0)
    {l=R;
     while(l->grand->grand !=0) l=l->grand;
     delete l->grand;l->grand=0 ;
    }
   else
    {l=R ;
      R=R->petit; delete l;
     }
}
}
```

b)

```
const int DimMax=.. ;
const int snil=0 ;
     struct sommet
     {    int cle;
          int petit,grand ;
     };

     Struct pile
     {int contenu[DimMax] ;
      int sommet ;
     } ;

     sommet groupsom[DimMax] ;
```

Appel à la fonction :

```
     affichFeuilIter(R) ;
```

Fonction :

```
void affichFeuilIter(arbre * R)
{
     pile p ;
```

```
        init(p) ;
        arbre * courant = R ;
        while((courant !=0) || (!vide(P))
        {
            if(courant==snil)
              depiler(p,courant);

             else while(groupsom[courant].petit !=snil)
              {empiler(p,courant) ;
               courant= groupsom[courant].petit ;
              }

        if ((groupsom[courant].petit==0)&&(
        groupsom[courant].grand=0))
        cout<< groupsom[courant].cle<< endl ;


        courant = groupsom[courant].grand ;
        }

}
```