المعهد الوطني للعلوم الفيزيائية والتكنولوجيا
nsat

Devoir surveillé ☐	Examen ☒	Session: principale ☒ de contrôle ☐

Matière : **Algorithmique**
Enseignant(s) : Riadh ROBBANA
Filière(s) : **MPI**
Barème : Ex 1 : 8 pts, Ex 2 : 6 pts et Ex 3 : 6 pts
Nombre de pages : Deux page

Semestre: Deuxième
Date: 2017
Durée: 1h 30 mn
Documents: autorisés ☐ non autorisés ☒

Exercice 1 (8 points)

Soit la structure de liste suivante :

```
struct Noeud
{
  int val ;
  struct Noeud *next ;
} ;
typedef struct Nœud* LISTE ;
```

On se propose de compresser une liste chaînée triées. La compression se fera en remplaçant chaque séquence du même élément par un seul nœud contenant l'élément et son nombre d'apparitions. Un seul parcours de la liste à compresser est permis.

1- Proposer la structure de la liste compressée LISTE_C.
2- Écrire la fonction compression qui permet de compresser une liste
3- Écrire la fonction décompression qui, à partir de la liste compressée récupère la liste décompressée.

Exercice 2 (6 points)

Les fonctions relatives aux files et piles vues en cours peuvent être utilisées.

1. Écrire une fonction C qui prend en argument une pile P et renvoie une nouvelle pile P0 contenant les mêmes éléments que P mais rangés dans l'ordre inverse. (Le sommet de P0 est l'élément au fond de P,..., et le sommet de P se retrouve au fond de P0).

NB : Vous ne pouvez utiliser que les PILES P et P0.

2. Écrire une fonction C qui prend en argument une file F et renvoie une nouvelle file F0 contenant les mêmes éléments que F mais rangés dans l'ordre inverse. (Le premier élément de F 0 est le dernier élément de F, ..., et le premier de F se retrouve le dernier de F0).

NB : Vous ne pouvez utiliser que les FILES F et F0.

Exercice 3 (6 points)

Soit la structure d'arbre suivante

```
struct abr
{
  Int val ;
  struct abr *gauche ;
  struct abr *droite ;
} ;
```

On suppose que lors de l'insertion d'un nœud dans un arbre binaire on a pu avoir les éléments ayan
une même clé successivement sur la branche gauche comme le montre l'exemple. C'est à dire que lor
de l'insertion d'une clé existante, ce nœud sera placé directement comme fils gauche du premier nœu
ayant la même clé.
Nous souhaitons éliminer les redondances d'un élément donné pour n'en garder qu'un seul. Il s'agit de
pointer sur un élément d'une clé donnée et de supprimer tous les autres éléments ayant la même clé e
que l'on trouve de manière successive sur un chaînage gauche.

```
void  supprime_redondance(ABR *racine, int cle)
{
}
```

L'exemple illustre l'arbre initial et le résultat après la suppression des redondances de la clé 6.