Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université de Carthage
Institut National des Sciences Appliquées et de Technologie

Université de Carthage

Devoir Surveillé ☐	Examen ■	Session principale ■	
		Session de contrôle ☐	
Matière :	Algorithmique et structures de données I	**Semestre :** 1	
Enseignant(s) :	Majdi Jribi et Hazar Mliki	**Date:**11 Janvier 2023	
Filière(s) :	MPI	**Durée:** 1h30	
Barème :	20 pts	**Documents :** autorisés ☐	
Nombre de pages :	02	non autorisés ■	

<u>Remarque importante</u> : la déclaration et l'utilisation de tableaux de différents types sont strictement interdites

<u>Problème</u> (20 pts)

Une liste d'éléments est un ensemble d'éléments repérés par leur rang.
Exemple:
La liste $L = \{5, 9, 14, 6, 3\}$
L'élément **5** est le premier élément de la liste L. Son rang est 1.
L'élément **9** est de rang **2** (deuxième élément de la liste L) et ainsi de suite pour les autres éléments.

Une liste peut être modélisée par une implémentation à base de tableaux par la structure de données suivante :

```
typedef struct{
int * T;
int nb;
int Tmax;
} Liste_Tab ;
```

- Le champ **T** est un tabeau d'entier contenant les éléments de la liste.
- Le champ **nb** est le nombre d'éléments dans la liste et par conséquent dans le tableau **T**.
- Le champ **Tmax** est la taille maximale du tableau **T**.

Pour la liste $L = \{5, 9, 14, 6, 3\}$, elle est modélisée par la structure de données **Liste_Tab** comme suit :

- Le tableau **T** :

5	**9**	**14**	**6**	**3**

- **nb** vaut 5
- **Tmax** est égale par exemple à 50

1- Ecrire, en langage C, la fonction **void Creation_Liste (Liste_Tab * L)** qui permet de créer la liste vide. Le champ **Tmax** est lu par l'utilisateur et les **Tmax** cases du tableau T sont allouées dynamiquement.

2- Ecrire, en langage C, la fonction **int dernier_elt(Liste_Tab L)** qui permet de retouner le dernier élément de la liste **L.**

3- Ecrire, en langage C, la fonction **Liste_Tab Ajout_Debut(Liste_Tab L, int x)** qui permet d'ajouter l'élément entier **x** au début de la liste **L.**

4- Ecrire, en langage C, la fonction **Liste_Tab Supprimer_Fin(Liste_Tab L)** qui permet de supprimer le dernier élément de la liste **L.**

5- On suppose maintenant qu'on ne peut qu'ajouter au début de la liste (la fonction **Liste_Tab Ajout_Debut(Liste_Tab L, int x)**), supprimer que le dernier élément de la liste (La fonction **Liste_Tab Supprimer_Fin(Liste_Tab L)**) et on n'a accès qu'au dernier élément de la liste (la fonction **int dernier_elt(Liste_Tab L)**)pour faire les traitement sur la liste.

Exemple : Pour $L = \{5, 9, 14, 6, 3\}$
Si on demande d'accéder au troisième élémnet de la liste (ici 14), il faut procéder comme suit :
L=Supprimer_Fin(L) : ici 6 devient le dernier élément de la liste.
L=Supprimer_Fin(L) : ici 14 devient le dernier élément de la liste.
y=dernier_elt(L) : on trouve dans **y** la valeur 14 demandée.

5.1. Ecrire, en langage C, la fonction **Liste_Tab Ajout_pos(Liste_Tab L, int x, int pos)** qui permet d'ajouter l'élément entier **x** à la position **pos.**

5.2. Ecrire, en langage C, la fonction **Liste_Tab Supprimer_pos(Liste_Tab L, int pos)** qui permet de supprimer l'élément qui se trouve à la position **pos.**

5.3. Ecrire, en langage C, la fonction **void Copie_inversée(Liste_Tab L1, Liste_Tab* L2)** qui permet de faire une copie inversée de **L1** dans **L2**

6- On suppose qu'on dispose des fonctions :

- **int premier_elt(Liste_Tab L)** qui permet de retourner le premier élément de la liste L.
- **Liste_Tab Supprimer_premier(Liste_Tab L)** qui permet de supprimer le premier élément de la liste.

On suppose maintenant qu'on ne peut qu'ajouter au début de la liste (la fonction **Liste_Tab Ajout_Debut(Liste_Tab L, int x)**), supprimer que le premier élément de la liste (la fonction **Liste_Tab Supprimer_premier(Liste_Tab L)**)et on n'a accès qu'au premier élément de la liste (la fonction **int premier_elt(Liste_Tab L)** pour faire les traitement sur la liste.

Ecrire, en langage C, la fonction **void Copie(Liste_Tab L1, Liste_Tab * L2)** qui permet de faire une copie de **L1** dans **L2. L'ordre d'apparition des éléments doit être respecté.**