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

Devoir Surveillé ☐	Examen ■	Session principale	■
		Session de contrôle	☐
Matière :	Algorithmique et structures de données II	**Semestre :** 2	
Enseignant(s) :	Majdi Jribi et Rabaa Youssef	**Date:** 23 Mai 2022	
Filière(s) :	MPI	**Durée:** 1h30	
Barème :	10+10	**Documents :** autorisés	☐
Nombre de pages :	03	non autorisés	■

Toutes les réponses doivent être rédigées en langage C

Exercice 1

Nous voudrions informatiser la gestion des files de véhicules dans une station de péage. Un véhicule est caractérisé par un numéro d'immatriculation (entier) et une marque (chaîne de caractères).

La file de véhicule qu'on nommera *file_vehicule* est caractérisée par la tête de la file et la queue de la file.

Soient les structures de données correspondantes :

```
typedef struct vehicule {
        int num ;
        char marque[50] ;
        struct vehicule *suivant,
} Vehicule;

typedef struct {
        Vehicule *tete;
        Vehicule *queue;
} file_vehicule;
```

1. Ecrire les fonctions C ***void créer_file (file_vehicule *fv)*** et ***int file_vide (file_vehicule fv)*** qui permettent respectivement de créer une file de véhicules vide et de tester si une file de véhicules est vide.
2. Ecrire la fonction C ***void enfiler(file_vehicule *fv, int val, char *mv)*** qui permet d'enfiler un véhicule dans la file de véhicules en précisant son numéro et sa marque.
3. Ecrire la fonction C ***void defiler (file_vehicule *fv)*** qui permet de défiler un véhicule de la file de véhicules.

Une station de péage est considérée comme une liste de guichets où chaque guichet est caractérisé par un numéro et par la file de véhicules qui se trouve devant ce guichet.
Soient les définitions relatives à la structure de données guichet et à la liste simplement chaînée de guichets :

```
typedef struct guichet {
        int numg ;
        file_vehicule *fv ;
        struct guichet *suivant,
} Guichet;

typedef Guichet * Liste_guichet;
```

4. On vous demande d'écrire la fonction C ***void ajouter_vehicule(Liste_guichets *LG, int num, vehicule *v)*** qui permet d'ajouter un véhicule v au guichet numéro **num**.
   Il faudra d'abord chercher dans la liste le guichet de numéro **num**. Si le numéro n'existe pas, un message d'erreur doit être renvoyé. Sinon, le véhicule devra être enfilé à la file de véhicules correspondant au guichet.

Exercice 2

On dispose d'un fichier « **manifestations.txt** » contenant la liste de manifestations organisées par un organisateur. Chaque ligne de ce fichier contient les informations suivantes :

Identifiant	: 4 Caractères (entier)
Nom de la manifestation	: 20 caractères (chaine de caractères)
Date	: 8 caractères (chaines de caractères)
Nombre de spectateurs	: 4 caractères (entier)
Bénéfices	: 8 caractères (réel à trois chiffres après la virgule)

Une manifestation est désignée par son identifiant. Une manifestation avec le même identifiant peut exister plusieurs fois dans le fichier « **manifestations.txt** »

1. Ecrire la fonction **void Affiche_Manifestations(FILE* fp, int nb_spectateurs)** qui permet d'afficher les noms et les dates de spectacles dont le nombre de spectateurs est supérieur à **nb_spectateurs.**
2. Ecrire la fonction **int Mise_à_jour_Manif(FILE* fp, int id, char* date, float benefice)** qui permet de mettre à jour le fichier « **manifestations.txt** » en modifiant la valeur de bénéfices par **benefice** d'un spectacle désigné par un identifiant **id** et pour une date donnée **date.**

La fonction devra retourner 1 si le logiciel a été modifié, 0 sinon.

3. Soit la structure de données suivante :

```
typedef struct {
int id ;
char nom_manifestation[21];
float total_benefices ;
} Manifestation ;
```

Le champ **total_benefices** représente la somme des benefices d'un spectacle désigné par son identifiant **id**.
On se propose de créer un nouveau fichier texte « **Manif_Org.txt** ».

Chaque ligne de ce fichier contient les informations suivantes :

Identifiant	: 4 Caractères (entier)
Nom de la manifestation	: 20 caractères (chaine de caractères)
Total de bénéfices	: 8 caractères (réel à trois chiffres après la virgule)

Ecrire la fonction **void Org_Manifestations(FILE* fp)** qui à partir du fichier « **manifestations.txt** », permet de créer le nouveau fichier « **Manif_Org.txt** » dans lequel toutes manifestations ayant le même identifiant **id** doivent être enregistrées dans la même ligne. La partie **total_benefices** correspond à la somme de tous les benefices relative à l'identifiant **id.**