NF16 - TP 3 – Listes chaînées

Ce TP est basé sur l'exercice 4 du médian de NF16 proposé au printemps 2011. Il a pour objectif de se familiariser avec les listes chaînées et les différentes opérations nécessaires pour les manipuler.

On cherche à développer un outil informatique pour la gestion des notes des étudiants d'une université.

A. Structures

1. Définir la structure **Note** et le type correspondant **T_Note**

La structure devra entre autres contenir les champs suivants :

- La note obtenue, nombre réel compris entre 0 et 20 - *de type float*
- La matière concernée par l'épreuve, de type chaîne de caractères

2. Définir le type **T_ListeNotes** - *de type pointeur vers une structure Note* - qui servira à représenter la liste des notes obtenues par un étudiant donné

3. Définir la structure **Etudiant** et le type correspondant **T_Etudiant**

La structure devra entre autres contenir les champs suivants :

- L'identifiant unique de l'étudiant - *de type int*
- Le nom de l'étudiant, de type chaîne de caractères
- Le prénom de l'étudiant, de type chaîne de caractères
- La liste des notes obtenues par l'étudiant - *de type T_ListeNotes*
- Le nombre de notes obtenues par l'étudiant - *de type int*
- La moyenne de toutes les notes obtenues par l'étudiant - *de type float*

4. Définir le type **T_ListeEtu** - *de type pointeur vers une structure Etudiant* - qui servira à représenter une liste d'étudiants. La liste doit être triée par identifiant unique de l'étudiant !

B. Fonctions requises

1. Création d'un élément d'une liste de notes :

```
T_Note *creerNote(float note, char *matiere)
```

2. Création d'un élément d'une liste d'étudiants :

```
T_Etudiant *creerEtudiant(int idEtu, char *nom, char *prenom)
```

3. Ajout d'une note en tête d'une liste de notes (renvoie la liste de notes modifiée) :

```
T_ListeNotes ajouterNote(float note, char *matiere, T_ListeNotes listeNotes)
```

4. Ajout d'une note pour un étudiant dans une liste d'étudiants :

```
T_ListeEtu ajouterNoteEtu(float note, char *matiere, int idEtu, T_ListeEtu listeEtu)
```

Cette fonction renvoie la liste d'étudiants modifiée. On veillera à bien gérer le cas de l'ajout d'un étudiant dans la liste si celui-ci n'y figure pas encore.

5. Suppression d'une note pour un étudiant dans une liste d'étudiants :

```
T_ListeEtu supprimerNoteEtu(char *matiere, int idEtu, T_ListeEtu listeEtu)
```

Cette fonction renvoie la liste d'étudiants modifiée. On veillera à retirer l'étudiant de la liste si celui-ci n'a plus aucune note.

6. Affichage d'une liste d'étudiants (nom, prénom, liste des notes) :

```
void afficherListeEtu(T_ListeEtu listeEtu)
```

7. Affichage du classement des étudiants par ordre décroissant de la moyenne (nom, prénom, moyenne) :

```
void afficherClassement(T_ListeEtu listeEtu)
```

8. Ecrire une procédure **sousListes** qui, étant données une liste d'étudiants **T_ListeEtu listeEtu** et une matière **char *matiere**, permet d'obtenir trois listes d'étudiants caractérisées comme suit :
   - Liste des étudiants de **listeEtu** ayant réussi l'épreuve
   - Liste des étudiants de **listeEtu** ayant raté l'épreuve
   - Liste des étudiants de **listeEtu** n'ayant pas passé l'épreuve

C. Programme Principal :

Utiliser les fonctions précédentes pour proposer à l'utilisateur le menu interactif suivant :

1. Créer la liste d'étudiants
2. Ajouter une note pour un étudiant
3. Supprimer une note pour un étudiant
4. Afficher la liste des étudiants
5. Afficher le classement
6. Afficher les sous-listes pour une épreuve
7. Quitter

Consignes générales :

- **Sources**

À la fin du programme, les blocs de mémoire dynamiquement alloués doivent être proprement libérés. Vous devrez également être attentifs à la complexité des algorithmes implémentés.

L'organisation MINIMALE du projet est la suivante :
- Fichier d'en-tête tp3.h, contenant la déclaration des structures/fonctions de base,
- Fichier source tp3.c, contenant la définition de chaque fonction,
- Fichier source main.c, contenant le programme principal.

- **Rapport**

Votre rapport de quatre pages maximum contiendra :

- La liste des structures et des fonctions supplémentaires que vous avez choisi d'implémenter et les raisons de ces choix.
- Un exposé succinct de la complexité de chacune des fonctions implémentées.

Votre rapport et vos fichiers source feront l'objet d'une remise de devoir sur Moodle dans l'espace qui sera ouvert à cet effet quelques jours suivant votre démonstration au chargé de TP (un seul rendu de devoir par binôme).