Programmation C avancée : Exercice de rappels

Cet exercice permet de récapituler l'ensemble des concepts abordés les années précédentes et vérifier de manière expérimentale qu'ils sont acquis :

- entrées / sorties
- structures
- tableaux
- chaînes de caractères
- pointeurs
- allocations dynamiques
- fichiers
- compilation séparée / protection contre multiple inclusion
- Makefile

Il est recommandé de créer un module séparé pour :

- les tableaux de notes
- les matières
- les étudiants
- les fonctions d'affichage
- les fonctions de sauvegarde et de restauration des données
- les fonctions qui récupèrent les données du fichier `data.txt`
- les fonctions qui effectuent des tris de données
- toute autre fonctionnalité que vous pourriez rajouter

Votre programme doit respecter les consignes suivantes :

- compilation avec un Makefile le plus générique possible
- présence d'un fichier README qui indique comment compiler et utiliser le programme
- ajout de commentaires dans chaque fichier source au format Doxygen
- génération de la documentation possible grâce à une cible du Makefile
- détection de problème et retour de valeur spécifique pour traitement par les fonctions appelantes
- utilisation de symboles en anglais exclusivement
- libération de toute la mémoire consommée à partir du moment où elle n'est plus utile

.

Format du fichier de données

Soit un fichier (fourni en TD) `data.txt` qui contient des données pédagogiques d'étudiants. Le format est le suivant :

```
ETUDIANTS
numero;prenom;nom;age
226345678;Alexander;Müller;19
226987654;Clara;Rossi;21
226112233;Felix;Dubois;20
226556789;Isabella;Ivanov;18
...
```

Puis dans ce même fichier, après les étudiants (dont le nombre n'est pas précisé explicitement), il peut y avoir des lignes vides puis une liste de matières :

```
MATIERES
nom;coef
Mathematiques;2.25
Physique;1.75
Informatique;2.5
Chimie;1.5
...
```

Enfin, après les matières, il peut y avoir des lignes vides suivies des notes des étudiants, dans un ordre totalement aléatoire avec le format :

```
NOTES
id;nom;note
226767522;Geographie;18.3
226112457;Mathematiques;2.2
226334459;Histoire;1.6
226778463;Histoire;13.1
226556461;Mathematiques;19.4
226778899;Economie;9.8
226989484;Physique;2.0
...
```

Chaque ligne ici contient l'identifiant unique d'un étudiant, suivi du nom unique d'une matière, suivi d'une valeur réelle qui est la note obtenue par l'étudiant pour cette matière. Dans chaque matière, toutes les notes ont le même poids.

Question 1

1. Ecrire la structure `Grades` permettant de stocker plusieurs valeurs réelles.
   Cette structure doit contenir le pointeur vers une zone contenant les données (tableau, liste chaînée) ainsi que la taille de cette zone.
2. Ecrire la structure `Course` qui contiendra plusieurs notes (`Grades`), ainsi que le nom de la matière (alloué dynamiquement au plus juste), son coefficient, et la moyenne de cette matière.
3. Ecrire la structure `Student` qui intègre l'identifiant unique de l'étudiant, son prénom et son nom, tous deux alloués dynamiquement au plus juste, son age, un tableau ou une liste chaînée de matières (`Course`) avec sa taille, ainsi que la moyenne générale de l'étudiant. On ne comptera pas une matière qui ne possède aucune note dans le calcul de la moyenne générale.
4. Ecrire la structure `Prom` qui contiendra l'ensemble des informations de la promotion d'étudiants, ainsi que le nombre d'étudiants.

   Toutes ces structures doivent prendre le minimum de place en mémoire. Il faut donc revoir l'ordre des membres si cela permet de gagner en mémoire.
   Pour chacune de ces structures, écrire les fonctions `constructeur` et `destructeur` pour avoir les services bas-niveau nécessaires pour la suite.

Question 2

1. Ecrire les fonctions nécessaires pour récupérer à partir du fichier de données une ligne d'une matière, d'un étudiant, ou d'une note, et la stocker en mémoire. Ces fonctions prennent au moins en paramètre une chaine de caractères (la ligne provenant du fichier).
   Ces fonctions peuvent soit allouer une zone en mémoire, la remplir et la retourner, soit prendre la zone mémoire de destination et directement remplir.
   A chaque ajout d'une note en mémoire, les moyennes doivent être mises à jour.
2. Ecrire la fonction permettant de stocker en mémoire l'ensemble de la promotion (tableau ou liste de `Student`) à partir du chemin du fichier de données.
3. Ecrire la ou les fonction(s) permettant d'afficher l'ensemble des informations contenues dans la promotion stockée en mémoire (informations personnelles, listes des matières et des notes, moyennes par matière et moyenne générale).

Question 3

1. Ecrire une fonction qui sauvegarde l'ensemble de la promotion en mémoire dans un fichier au format binaire.
2. Ecrire la fonction associée pour restaurer en mémoire la promotion sauvegardée initialement dans le fichier binaire.

   Le format exact du fichier binaire est laissé libre.
   Le programme doit être capable de sauvegarder son contexte mémoire dans un fichier, puis, après redémarrage, recharger ce contexte et l'afficher pour prouver que la fonctionnalité de sauvegarde/restauration est opérationnelle.

Question 4

1. Ecrire une fonction qui va prendre en entrée une promotion, et retourner un tableau contenant, au plus, les 10 meilleurs étudiants (ceux ayant la plus grande moyenne, triés dans l'ordre décroissant).
2. Ecrire une fonction qui prend en entrée le nom d'une matière et la promotion d'étudiants, et qui va retourner les 3 meilleurs étudiants de cette matière (tableau de 3 étudiants alloué dynamiquement)