Contexte

Le but de ce TP est de développer une mini-application de gestion des étudiants de première année. Cette application doit permettre :

- De gérer une liste d'étudiants (ajouter/supprimer un étudiant)
- D'effectuer une recherche selon différents critères (matricule, nom, statuts …etc.)
- D'afficher convenablement les différents résultats.
- Sauvegarder/récupérer les données dans/depuis un fichier.

Les étudiants de première année ont trois matières principales : mathématique, physique et informatique de coefficients respectifs 2,2 et 1. Ils doivent également effectuer un stage ouvrier de trois semaines ainsi qu'un minimum de huit heures de monitorat (participer aux journées portes ouvertes de l'école, assister un enseignant lors de séances de TP …etc.).

Afin de valider son année, un étudiant doit n'avoir aucune note éliminatoire (inférieure à 5) et soit obtenir une moyenne générale supérieure ou égale à 10 et avoir validé son stage ainsi que ses heures de monitorat, soit obtenir une moyenne générale supérieur ou égale à 12 et avoir validé au minimum son stage.

Un étudiant est décrit par son numéro d'étudiant (son matricule compris entre 1001 et 1999), son nom, son prénom, ses notes, le statut de son stage (validé ou non) ainsi que le statut de son monitorat.

Objectif du TP

La réalisation d'un programme complexe peut s'avérer beaucoup plus simple lorsque ce dernier est découpé en plusieurs petits sous-programmes indépendants (« diviser pour mieux régner »). Bien sûr, il est nécessaire d'avoir une bonne vision globale des fonctionnalités à implémenter ainsi que du déroulement de plusieurs scénarios possible d'exécution afin de permettre un découpage intelligent.

Dans le cas de ce TP le découpage a déjà été imaginé. L'idée est de coder des petits sous-programmes autonomes et de les tester **au fur et à mesure** pour enfin les intégrer dans une application globale qui réponds aux spécifications souhaitées.

Tout d'abord, commencez par ouvrir le projet ***TP2*** que vous pouvez télécharger sur openschool. Compilez-le et exécutez-le, vous pourrez vous balader dans un menu textuel ce qui vous permettra d'avoir une bonne vision globale de la manière dont le programme a été découpé. Le projet contient déjà quelques sous-programmes, notamment ceux qui permettent de gérer l'interface (le menu). Vous devez, au fil des questions, compléter le programme avec les différentes fonctions manquantes. Une fois vos fonctions testées, localisez dans le code, les différents emplacements ou insérer l'appel de vos sous-programmes.

1 Structure de données

1 Ecrire le code permettant de définir un nouveau type ***Etudiant*** respectant les spécifications précédentes.

2 Sachant que la promotion compte au maximum 300 étudiants, écrire le code permettant de définir une structure de donnée adaptée pour stocker les données de tous les étudiants. (Nous ferons en sorte, par la suite, que les étudiants soient triés par ordre alphabétique selon le nom).

2 Utilisation des fichiers

Afin de vous éviter d'avoir à remplir manuellement votre tableau d'***Etudiant***, vous pouvez avoir recours à la fonction :

```
int lire_fichier( Etudiant tab[])
```

Cette fonction permet de remplir le tableau ***tab*** à partir des données lues dans un fichier, elle retourne le nombre d'étudiants stockés dans le tableau.

Vous trouverez également le sous-programme :

```
void ecrire_fichier( Etudiant tab[], int nb)
```

Qui écrit dans un fichier les données à partir d'un tableau ***tab*** contenant ***nb*** étudiants.

Ces deux fonctions sont déjà implémentées. Vous pouvez donc les appeler dans votre main afin de disposer d'un tableau d'étudiants rempli sur lequel vous pourrez tester vos fonctions.

3 Affichage (affichage.h et affichage.c)

3 Ecrire un sous-programme prenant en entrée un paramètre de type ***Etudiant*** de prototype :

```
void afficher_etudiant(Etudiant e);
```

Et qui affiche les données relatives à cet étudiant selon le format suivant :

```
Matricule  Nom              Prénom           Math   Phy    Info     Stage    Monitorat    Moy     Statut
-------------------------------------------------------------------------------------------------------
1001       ABDERREZAK       Wissem           18.00  4.00   4.00     0        0            9.60    Ajourné
```

4 Ecrire un sous-programme prenant en paramètre un tableau d'***Etudiant*** ainsi que sa taille de prototype :

```
  void afficher_list(Etudiant tab[], int nb);
```

Et qui affiche les données de ce tableau selon le format suivant :

```
---------------------------------------    180 étudiants    ---------------------------------------------------
Matricule  Nom              Prénom              Math   Phy    Info    Stage   Monitorat   Moy     Statut
-------------------------------------------------------------------------------------------------------------
1001       ABDERREZAK       Wissem              18.00  4.00   4.00    0       0           9.60    Ajourné
1002       ADJIVON          Hugo                15.00  7.00   20.00   0       0           12.80   Ajourné
1003       AERNOUTS         Louis               0.00   7.00   17.00   0       1           6.20    Ajourné
1004       AILLOT           Paul                9.00   20.00  15.00   1       0           14.60   Admis
1005       AJAZ-HAIDER      Ahsan               8.00   16.00  11.00   0       1           11.80   Ajourné
1006       ALEXANDRE        Emmanuel            16.00  9.00   1.00    0       0           10.20   Ajourné
1007       ALFRED           Nicolas             15.00  12.00  7.00    0       0           12.20   Ajourné
1008       ALVES            Tom                 1.00   12.00  6.00    1       0           6.40    Ajourné
1009       ANDRIEUX         Cyprien             16.00  1.00   14.00   1       1           9.60    Ajourné
1010       AOUAD            Youssef             19.00  2.00   12.00   1       0           10.80   Ajourné
1011       APIED            Quentin             5.00   12.00  6.00    0       1           8.00    Ajourné
1012       ARCHAMBAUD       Damien              4.00   3.00   11.00   0       1           5.00    Ajourné
1013       ARMAND           Marius              3.00   18.00  18.00   1       0           12.00   Ajourné
1014       ARNAL            Paul                9.00   2.00   20.00   0       0           8.40    Ajourné
1015       ARNAUD           Maxime              15.00  8.00   15.00   0       1           12.20   Admis
```

4 Calcul (calcul.h et calcul.c)

5 Ecrire un sous-programme prenant en entrée un paramètre de type ***Etudiant***, qui calcule la moyenne de cet étudiant. La moyenne calculée est retournée par le sous-programme. Prototype :

```
float calcul_moyenne(Etudiant e);
```

6 Ecrire un sous-programme prenant en entrée un paramètre de type ***Etudiant*** et qui retourne 1 si l'étudiant est admis, 0 sinon, de prototype :

```
int calcul_statut(Etudiant e);
```

5 Sous-programmes de mise-à-jour des données (maj.h et maj.c)

Dans cette partie, il s'agit de coder l'ensemble des sous-programmes servant à modifier les données.

7 Ecrire un sous-programme qui permet à l'utilisateur de saisir les données d'un étudiant respectant le prototype suivant :

```
Etudiant saisir_etudiant();
```

La moyenne ainsi que le statut (Ajourné ou Admis) ne sont pas saisis par l'utilisateur mais calculés et renseignés automatiquement par le sous-programme.

8 Ecrire un sous-programme permettant d'insérer un nouvel étudiant dans un tableau considéré comme étant déjà trié. Attention, le tableau doit rester bien ordonné après insertion. Prototype d la fonction :

```
void ajouter_etudiant(Etudiant tab[], int nb, Etudiant e);
```

9 Ecrire un programme permettant de supprimer un étudiant d'un tableau ordonné, à partir de son numéro d'étudiant. Le sous-programme devra retourner 1 si l'étudiant a bien été supprimé, 0 sinon.

```
int supprimer_matricule(Etudiant Tab1[],int nb, int mat)
```

6 Sous-programmes de consultation des données (consulter.h et consulter.c)

10 Ecrire un sous-programme qui recherche un étudiant dans un tableau à partir de son matricule. Ce sous-programme retournera l'étudiant trouvé s'il existe. Dans le cas contraire, elle retournera un étudiant avec un matricule égale à 0.

```
Etudiant recherche_matricule(Etudiant Tab1[],int nb);
```

11 Ecrire un sous-programme respectant le prototype suivant :

```
int filtre_nom(char *name, Etudiant Tab1[],int nb1,Etudiant Tab2[]);
```

qui extrait dans un tableau ***Tab2*** à partir du tableau ***Tab1*** (de taille ***nb1***) tous les étudiants qui ont leurs noms égaux à ***name***. Cette fonction retournera le nombre d'étudiants extraits.

12 Ecrire un sous-programme respectant le prototype suivant :

```
int filtre_statut(int s,Etudiant Tab1[],int t1,Etudiant Tab2[]);
```

qui extrait dans ***Tab2*** :

- Les étudiants ajournée si ***s***=0.
- Les étudiants admis sinon.

13 Ecrire un sous-programme prenant en entrée un tableau d'***Etudiant*** et qui retourne le major de la promotion (Attention celui-ci doit, bien évidemment, être Admis). Prototype :

```
Etudiant recherche_major(Etudiant Tab1[],int nb);
```

7 Questions bonus

14 Ecrire un sous-programme respectant le prototype suivant :

```
int tri_moy(Etudiant Tab1[],int t1,Etudiant Tab2[]);
```

qui tri, par ordre décroissant selon la moyenne, dans un tableau ***Tab2*** le contenu du tableau ***Tab1*** de taille ***t1***.

15 Ecrire un sous-programme respectant le prototype suivant :

```
int tri_math(Etudiant Tab1[],int t1,Etudiant Tab2[]);
```

qui tri, par ordre décroissant selon la note de mathématique, dans un tableau ***Tab2*** le contenu du tableau ***Tab1*** de taille ***t1***.

16 Ecrire un sous-programme respectant le prototype suivant :

```
int tri_phy(Etudiant Tab1[],int t1,Etudiant Tab2[]);
```

qui tri, par ordre décroissant selon la note de physique, dans un tableau ***Tab2*** le contenu du tableau ***Tab1*** de taille ***t1***.

17 Ecrire un sous-programme respectant le prototype suivant :

```
int tri_inf(Etudiant Tab1[],int t1,Etudiant Tab2[]);
```

qui tri, par ordre décroissant selon la note d'informatique, dans un tableau ***Tab2*** le contenu du tableau ***Tab1*** de taille ***t1***.

18 Ecrire un sous-programme qui calcule et qui affiche les statistiques de la promotion selon le format suivant :

```
---------------------------------------    180 étudiants    ---------------------------------------------------
                                  Mathématique   Physique    Informatique    générale
-------------------------------------------------------------------------------------------------------------
Moyenne                           9.99           9.62        11.04           10.05
Min                               0.00           0.00        0.00            1.20
Max                               20.00          20.00       20.00           18.80
-------------------------------------------------------------------------------------------------------------
pourcentage ayant eu la moyenne (%) 47.78         51.67       63.33           49.44

-------------------------------------------------------------------------------------------------------------
étudiants ayant validé leur stage     : 43.9 %
étudiants ayant validé leur monitorat : 48.3 %
étudiants ayant validé leur année     : 18.3 %
```