TRAVAIL PRATIQUE PERSONNEL : à rendre avant le 30 avril 2015 à 18H précises.

Le responsable des enseignements en Algorithme et Programmation 3 (unité d'enseignement organisée en tronc commun RTC_3 = **5** étudiants et GL_3 = **4** étudiants) veut évaluer la moyenne des seuls étudiants de la GL_3 sur la base de leurs notes de devoir CM = 30%, de projet TP = 20% et d'examen E = 50%.

Travail à faire :

1. Schématiser suivant une structure la table de gestion des données relatives à ce module d'enseignement.
2. Ecrire une fonction qui permet principalement :
   - de remplir cette base de données de l'effectif total d'étudiants au module.
   - de décider du passage pour tous les étudiants dont la moyenne est connue. (stocker dans un fichier)
3. Ecrire une fonction qui permet de déterminer la moyenne de la filière GL_3.
4. Déterminer un algorithme de tri qui classe les noms des étudiants au module par ordre alphabétique ; lequel code passera la liste achevée des étudiants à l'écran. (stocker dans le même fichier)
5. Prévoir une manière de permuter les positions des étudiants de la GL_3 ayant respectivement la plus forte moyenne et la plus faible. (stocker dans le même fichier) **TEST OBLIGATOIRE DANS LA FONCTION PRINCIPALE**

Exercice 1

(1)
```
void permutation (int a, int b)
{
   int c  ;
   c = a ;
   a = b ;
   b = c ;
}
```

(2)
```
void permutation (int* p_a, int* p_b)
{
   int c ;
   c = *p_a ;
  *p_a = b ;
   *p_b = c ;
}
```

1. Ecrire la formule d'appel de chacune des fonctions ci–dessous après avoir précisé leur objet.
2. Cerner la différence entre les bouts de codes (1) et (2).

Exercice 2

On se propose de manipuler dans ce programme les fractions ; étant entendu qu'une fraction s'affiche au format **a/b** avec a et b des entiers relatifs impérativement non nuls.

Travail à faire :

1. Ecrire une structure en rapport avec le contexte édicté.
2. Ecrire une fonction qui permet de n'entrer une fraction que selon la prescription ci-haut.
3. Ecrire une fonction qui permet de vérifier qu'une fraction est réductible ou pas.
4. Ecrire une fonction qui compare deux fractions.
5. Ecrire une fonction qui permet de tester si deux fractions sont opposées.
6. Ecrire une fonction qui permet de tester si deux fractions sont inverses.

Exercice 3

On souhaite concevoir une librairie pour la manipulation de matrices d'entiers. Une matrice est un tableau d'entiers a deux dimensions. On se propose donc d'utiliser la structure suivante pour représenter la matrice :

```
typedef struct {
                  int **valeurs ;
                  int longueur ;
                  int largeur ;
                } matrice ;
```

Travail à faire :

1. Ecrire la fonction **matrice creer_matrice ( )** qui demande à l'utilisateur de saisir deux entiers correspondant aux dimensions de la matrice, puis alloue le tableau a deux dimensions (en utilisant la fonction **malloc**).
2. Ecrire la fonction **matrice alea_matrice ( )** qui retourne une matrice remplie avec des entiers aléatoires compris entre 0 et 100 (à l'aide de la fonction **rand()**).
3. Ecrire la fonction **void affiche_matrice (matrice m)** qui affiche dans la console la matrice donnée en paramètre.
4. Ecrire la fonction **matrice addition_matrice (matrice mat1, matrice mat2)** qui effectue l'addition de deux matrices : MAT1 = MAT1 + MAT2
5. Ecrire la fonction **matrice entier_matrice (int x, matrice mat)** qui effectue la multiplication de la matrice MAT1 par un entier X : MAT1 = X * MAT1
6. Ecrire la fonction **matrice multiplication_matrice (matrice mat1, matrice mat2)** qui effectue le produit de deux matrices : MAT1 = MAT1 * MAT2

Travail pratique par groupe de 5 étudiants :

Les exercices 2 et 3 sont à présenter sous la forme d'une application qui intègre un menu et à rendre avant le 5 mai 2015 à 12H précises.