TD 9 : Les fichiers

1°) fichier texte. On dispose d'un très gros fichier texte contenant des coordonnées de point dans l'espace (3D) structuré de la manière suivante :

```
( 12.4, 32.1, -21 )
( 51, -3 .4, 18.3)
…
```

Les coordonnées d'un même point sont entre parenthèses et séparées par des virgules.

Ecrire le programme C qui lit ce fichier, calcule le centre de gravité de ces points (moyenne des x, moyenne des y et moyenne des z), construit le fichier texte de ces points en prenant le centre de gravité comme origine. On supposera le fichier initial sans erreurs.

2°) Fusion de fichier. Soit les déclarations suivantes :

```
typedef struct article
{ char designation[256] ;
  int reference ;
  int quantite ;
} Article ;
```

et soit f1 et f2 deux fichiers binaires constitués de tels enregistrements et triés suivant l'ordre croissant des références, écrire la fonction qui construit le fichier f3 (trié) qui est une fusion de f1 et f2.

3°) Tasser un fichier. Ecrire la fonction qui compacte le fichier précédant f3 en regroupant les enregistrements ayant la même référence et en cumulant les quantités.

4°) Un fichier image au format « pgm » permet de stocker des images en niveaux de gris et est structuré de la manière suivante :

- Entête de type texte
- Un octet par pixel (niveau de gris entre 0 et 255 codé en binaire)
- Les pixels sont rangés ligne par ligne, de haut en bas

```
p5
# quelques commentaires
# largeur et hauteur en mode texte :
256 200
# plus fort niveau de gris dans l'image :
255
suite de (largeur x hauteur) octets…
```

Ecrire la fonction lirePGM qui reçoit le nom d'un fichier pgm en argument et charge l'image dans une structure de donnée adaptée après l'allocation de l'espace nécessaire.

```
typedef unsigned char Pixel ;
typedef struct
{ int haut, larg ;
  Pixel *tab ;
} Image ;
Image I ;
```

Correction TD 9 : Les fichiers

1°)

```
#include <stdio.h>
FILE *F1, *F2 ;

int lirePoint(float *px, float *py, float *pz) /* retourne 0 si fin de
fichier */
{ char car ;
  if (fscanf(F1, "(%f,%f,%f)\n", px, py, pz) < 3) return 0 ;
  return 1 ;
}


int main(void)
{ float x0, y0, z0 ; /* coordonnées du barycentre */
  float x, y, z ;  /* coordonnées du point courant */
  int nbPts ; /* nbre de pts dans le fichier */

  F1 = fopen ("nuageDePoints.txt","r") ;
  if (F1 == NULL)
  { printf("erreur d'ouverture\n");
    return 0 ;
  }
  /* lecture de F1 pour le calcul du barycentre */
  x0 = y0 = z0 = 0 ;
  nbPts = 0 ;
  while (lirePoint(&x, &y, &z))
  { x0 = x0+x ;
    y0 = y0+y ;
    z0 = z0+z ;
    nbPts++;
  }
  x0 = x0/nbPts ;
  y0 = y0/nbPts ;
  z0 = z0/nbPts ;

  /* creation du fichier des pts centres */
  F2 = fopen ("nuageRecentre.txt","w") ;
  rewind(F1) ;
  while (lirePoint(&x, &y, &z))
    fprintf(F2, "(%7.2f, %7.2f, %7.2f)\n", x-x0, y-y0, z-z0) ;
  fclose(F1) ;
  fclose(F2) ;
  return 1;
}
```

2°)

Soit les déclarations suivantes :

```
typedef struct article
{
  int reference;
  int quantite ;
} Article ;
```

et soit f1 et f2 deux fichiers constitués de tels enregistrements et triés suivant l'ordre croissant des références,
écrire la fonction qui construit le fichier f3 (trié) qui est une fusion de f1 et f2.

```
int fusion(char n1[],char n2[],char n3[])
{ FILE *f1,*f2,*f3 ;
  Article e1,e2;
  f1 = fopen(n1,"r") ;
  if (f1 == NULL) return 0 ;
  f2 = fopen(n2,"r") ;
  if (f2 = NULL) { fclose(f1) ;return 0 ; }
  f3 = fopen(n3,"w") ;
  ok1 = fread(&e1,sizeof e1,1,f1) ;
  ok2 = fread(&e2,sizeof e2,1,f2) ;
  while (ok1 && ok2)
  { if (e1.reference < e2.reference)
    { fprint(e1,sizeof e1,1,f3) ;
      ok1 = fread(&e1,sizeof e1,1,f1) ;
    }
    else
    { fprint(e2,sizeof e2,1,f3) ;
      ok2 = fread(&e2,sizeof e2,1,f2) ;
    }
  }
  while (ok1)
  { fprint(e1,sizeof e1,1,f3) ;
    ok2 = fread(&e1,sizeof e1,1,f1) ;
  }
  while (ok2)
  { fprint(e2,sizeof e2,1,f3) ;
    ok2=fread(&e2,sizeof e2,1,f2) ;
  }
  fclose(f1) ;
  fclose(f2) ;
  fclose(f3) ;
  return 1 ;
}
```

3°)
Ecrire la fonction qui compacte le fichier précédant f3 en regroupant les enregistrements ayant la même référence et en cumulant les quantités. Le résultat sera un fichier f4 trié.

```
int compacter(char n1[],char n2[])
{ FILE *f1,*f2 ;
  Article e1,e2;
  f1 = fopen(n1,"r") ;
  if (f == NULL) return 0 ;
```

```
    f2 = fopen(n2,"w") ;
    if (!fread(&e2,sizeof e2, 1, f1)
    { fclose(f1);
      fclose(f2);
      return 1 ;
    }
    while (fread(&e1,sizeof e1, 1, f1)==1)
    { if (e2.reference == e1.reference)
        e2.quantite = e2.quantite + e1.quantite ;
      else
      { fprintf(e2,sizeof e2, 1, f2) ;
        e2 = e1 ;
      }
    }
    fprintf(e2,sizeof e2, 1, f2) ;
    fclose(f1)
    fclose(f2);
    return 1 ;
  }
```

4°) Ecrire la fonction lirePGM qui reçoit le nom d'un fichier pgm en argument et charge l'image dans une structure de donnée adaptée après l'allocation de l'espace nécessaire.

```
#define MAXLIGNE 80
Image LireImagePgm(char nom[], int marge)
{ Image image;
  FILE *f;
  char s[MAXLIGNE];
  int l, h, n, k /*, maxgrey */ ;
  unsigned char *ligne;

  /* Ouverture du fichier */
  f = fopen (nom, "r");
  if (f == NULL) exit(0); /* abandon */

  /* Lecture de l'entete */
  fgets (s, MAXLIGNE, f);
  if (s[0] != 'P' || s[1] != '5') exit(0);
  /* saut des commentaires */
  do
    fgets (s, MAXLIGNE, f);
  while (s[0] == '#');
  /* recuperer largeur et hauteur */
  sscanf (s, "%d %d", &(image.larg), &(image.haut));
  /* saut des commentaires */
  do
    fgets (s, MAXLIGNE, f);
  while (s[0] == '#');
  /* Lecture de maxgrey */
  /* sscanf (s, "%d", &maxgrey); */ /* on l'oublie */

  /* Allocation Image et lecture*/
```

```
  tailleImage = image.larg*image.haut*sizeof(Pixel) ;
  image.tab = malloc(tailleImage);
  if (fread(image.tab, tailleImage, f) != 1)
    exit(0); /* abandon */
  return image;
}
```