```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

typedef double** MAT1;

MAT1 allouMatrice1(int nl, int nc) { MAT1 m;
  int i;
  if ( (m = calloc(nl,sizeof(*m))) ==NULL) return NULL;
  if ( (*m = calloc(nl*nc,sizeof(**m))) ==NULL) { free(m); return NULL;}
  for(i=1;i<nl;i++) m[i]=m[i-1]+nc;
  return m;
}

double elementMatrice1(MAT1 m, int i, int j) {
  return m[i][j];
}

MAT1 sommeMatrice1(MAT1 m1, MAT1 m2, int nl, int nc) { MAT1 r;
  int i,j;
  if ( (r=allouMatrice1(nl,nc)) ==  NULL) return NULL;
  for (i=0; i<nl*nc; i++) r[0][i]=m1[0][i]+m2[0][i];
  return r;
}
/* Ou bien
MAT1 sommeMatrice1(MAT1 m1, MAT1 m2, int nl, int nc) { MAT1 r;
  int i,j:
  if ( (r=allouMatrice1(nl,nc)) ==  NULL) return NULL;
  for (i=0; i<nl; i++)
    for (j=0; j<nc; j++)
       r[i][j]=m1[i][j]+m2[i][j];
  return r;
}
*/

double minMatrice1(MAT1 m1, int nl, int nc) { double v;
  int i;
  for (v=m1[0][0], i=1; i<nl*nc; i++)
    if (m1[0][i] < v) v=m1[0][i];
  return v;
}

/* Ou bien
double minMatrice1(MAT1 m1, int nl, int nc) { double v=m1[0][0];;
  int i,j:
  for (i=0; i<nl; i++)
    for (j=0; j<nc; j++)
       if (m1[i][j]<v) v=m1[i][j];
  return v;
}
*/

MAT1 lectureMatrice1(char* fic, int* anl, int* anc) { MAT1 r;
  int i,j;
  FILE* fp;
  double val;
  if ( (fp=fopen(fic,"r"))==NULL) return NULL;
  fscanf(fp,"%d %d ",anl,anc);
  if ( (r=allouMatrice1(*anl,*anc)) ==  NULL) return NULL;
```



```c
  while ( (fscanf(fp,"%d %d %lf ",&i,&j,&val)==3))
     r[i][j]=val;
  fclose(fp);
  return  r;
}

void afficheMatrice1(MAT1 m, int nl,int nc) {
  int i,j;
  for (i=0; i<nl; i++) {
    for (j=0; j<nc; j++)
      printf("%5.2lf ",m[i][j]);
    puts("");
  }
}
/*
  Taille occupee = 4 + nl*4 + nl*nc*8
  Complecite : O(1)
*/

void  main1() { int nl,nc;
  MAT1 m1,m2,m3;
  m1=lectureMatrice1("data201705_1.dat",&nl,&nc);
  puts("Matrice m1"); afficheMatrice1(m1,nl,nc);
  m2=lectureMatrice1("data201705_2.dat",&nl,&nc);
  m3=sommeMatrice1(m1,m2,nl,nc);
  puts("Matrice m3"); afficheMatrice1(m3,nl,nc);
  printf("le min est : i%lf\n",minMatrice1(m1,nl,nc));
}


struct cellule2 {
    int col;
        double val;
    struct cellule2 * suiv;} ;

typedef struct cellule2 CHAINON2, * LISTE2;

typedef LISTE2* MAT2;

LISTE2 creerliste2(void) {  return NULL;    }

int estVideListe2(LISTE2 L) {   return !L;  }

LISTE2 ajoutListe2(int col, double val,  LISTE2 L){
    LISTE2 p=(LISTE2) calloc(1,sizeof(*p));
    if (p!=NULL) {
        p->val=val; p->col=col;
        p->suiv=L;
    }
    return p;
}


/* Cette version suppose que lig,col n'existe pas deja, ce qui est le cas du
    sujet.
 Si on permet les modification d'un lig,co existant, voici la bonne version


 LISTE2 ajoutListe2(int lig, double val,  LISTE2 L){
 LISTE2 p;
```

```
 p=valeurListe2(lig,L);
 if (p) { p->val=val; return L; }
 p=(LISTE2) calloc(1,sizeof(*p));
 if (p!=NULL) {
 p->val=val; p->col=col;
 p->suiv=L;
 }
 return p;
 }
 */

LISTE2 valeurListe2(int c, LISTE2 L){ LISTE2 p=L;
    while(!estVideListe2(p) && p->col!=c) p=p->suiv;
    return p;
}

void afficheListe2(LISTE2 L){ LISTE2 p=L;
    while(!estVideListe2(p)) {
        printf("%d %lf; ",p->col,p->val);
        p=p->suiv;
    }
}

MAT2 allouMatrice2(int n) { MAT2 m;
  if ( (m=calloc(n,sizeof(*m)))==NULL) return NULL;
  return m;
}

void ajoutMatrice2(MAT2 m, int i, int j, double val) {
  m[i]=ajoutListe2(j,val,m[i]);
}

double elementMatrice2(MAT2 m, int i, int j) {
  LISTE2 p;
  if ((p=valeurListe2(j,m[i]))==NULL) return 0;
  else return p->val;;
}

MAT2 sommeMatrice2(MAT2 m1, MAT2 m2, int nl, int nc) { MAT2 r;
  int i,j;
  double v;
  if ( (r=allouMatrice2(nl)) ==  NULL) return NULL;
  for (i=0; i<nl; i++)
    for (j=0; j<nc; j++) {
       v=elementMatrice2(m1,i,j) + elementMatrice2(m2,i,j);
       if (v!=0) ajoutMatrice2(r,i,j,v);
    }
  return r;
}

double minMatrice2(MAT2 m, int nl, int nc) { double val=0;
  LISTE2 p;
  int i;
  for (i=0; i<nl; i++)
    for (p=m[i]; !estVideListe2(p); p=p->suiv)
       if (p->val < val) val=p->val;
  return val;
}
```



```
MAT2 lectureMatrice2(char* fic, int* anl, int* anc) { MAT2 r;
  int i,j;
  FILE* fp;
  double val;
  if ( (fp=fopen(fic,"r"))==NULL) return NULL;
  fscanf(fp,"%d %d ",anl,anc);
  if ( (r=allouMatrice2(*anl)) ==  NULL) return NULL;
  while ( (fscanf(fp,"%d %d %lf ",&i,&j,&val)==3))
      ajoutMatrice2(r,i,j,val);
  fclose(fp);
  return  r;
}

void afficheMatrice2(MAT2 m, int nl,int nc) {
  int i,j;
  for (i=0; i<nl; i++) {
    for (j=0; j<nc; j++)
      printf("%5.2lf ",elementMatrice2(m,i,j));
    puts("");
  }
}
/*
  Taille occupee = 4+ nl*4 + k* (8+4+4)
  Complexite : o(k/nl)
  Pire cas : absence de l'elemetn o(K/nl)
  Amelioration simple: liste ordonnee par ordre croissant de colonnes
*/

void main2() { int nl,nc;
  MAT2 m1,m2,m3;
  m1=lectureMatrice2("data201705_1.dat",&nl,&nc);
  puts("Matrice m1"); afficheMatrice2(m1,nl,nc);
  m2=lectureMatrice2("data201705_2.dat",&nl,&nc);
  m3=sommeMatrice2(m1,m2,nl,nc);
  puts("Matrice m3"); afficheMatrice2(m3,nl,nc);
  printf("le mi est : %lf\n",minMatrice2(m1,nl,nc));
}


struct cellule3 {
        int lig,col;
        double val;
        struct cellule3 * suiv;} ;

typedef struct cellule3 CHAINON3, * LISTE3;

typedef LISTE3* MAT3;


int hash(int i, int t) {
    return (i/100+i%100)%t;
}

LISTE3 creerliste3(void) {      return NULL;    }

int estVideListe3(LISTE3 L) {   return !L;      }

LISTE3 valeurListe3(int l, int c, LISTE3 L){ LISTE3 p=L;
        while(!estVideListe3(p) && (p->col!=c || p->lig!=l) ) p=p->suiv;
        return p;
```

```
}

LISTE3 ajoutListe3(int lig, int col, double val,  LISTE3 L){
        LISTE3 p=(LISTE3) calloc(1,sizeof(*p));
        if (p!=NULL) {
                p->val=val; p->col=col; p->lig=lig;
                p->suiv=L;
        }
        return p;
}
/* Cette version suppose que lig,col n'existe pas deja, ce qui est le cas du
    sujet.
 Si on permet les modification d'un lig,co existant, voici la bonne version


LISTE3 ajoutListe3(int lig, int col, double val,  LISTE3 L){
    LISTE3 p;
    p=valeurListe3(lig,col,L);
    if (p) { p->val=val; return L; }
    p=(LISTE3) calloc(1,sizeof(*p));
    if (p!=NULL) {
        p->val=val; p->col=col; p->lig=lig;
        p->suiv=L;
    }
    return p;
}
*/

void afficheListe3(LISTE3 L){ LISTE3 p=L;
        while(!estVideListe3(p)) {
                printf("%d %d %lf; ",p->lig, p->col,p->val);
                p=p->suiv;
        }
}

MAT3 allouMatrice3(int n) { MAT3 m;
  if ( (m=calloc(n,sizeof(*m)))==NULL) return NULL;
  return m;
}

void ajoutMatrice3(MAT3 m, int t, int i, int j, double val) {
  m[hash(i,t)]=ajoutListe3(i,j,val,m[hash(i,t)]);
}

double elementMatrice3(MAT3 m, int t, int i, int j) {
  LISTE3 p;
  if ( (p=valeurListe3(i,j,m[hash(i,t)])) ==NULL) return 0;
  else return p->val;;
}

MAT3 sommeMatrice3(MAT3 m1, MAT3 m2, int t, int nl, int nc) { MAT3 r;
  int i,j;
  double v;
  if ( (r=allouMatrice3(t)) ==  NULL) return NULL;
  for (i=0; i<nl; i++)
    for (j=0; j<nc; j++) {
       v=elementMatrice3(m1,t,i,j) + elementMatrice3(m2,t,i,j);
       if (v!=0) ajoutMatrice3(r,t,i,j,v);
  }
  return r;
```



```
}

/*
  Valable si ordre total sur les listes avec lignes puis colonnes
*/
MAT3 sommeOptMatrice3(MAT3 m1, MAT3 m2, int t, int nl, int nc) { MAT3 r;
  int i,j;
  double v;
  if ( (r=allouMatrice3(t)) ==  NULL) return NULL;
    for (i=0; i<t; i++) {
      p1=m1[i]; p2=m2[i];
      if (p1!=NULL)
        if (p2!=NULL)
          if (p1->lig == p2->lig)
            if (p1->col == p2->col) {v=p1->val+p2->val; p1=p1->suiv; p2=p2->
                suiv; }
            else if (p1->col < p2->col) { v=v1->val; p1=p1->suiv; }
                 else { v=p2->val; p2=p2->suiv; }
       v=elementMatrice3(m1,t,i,j) + elementMatrice3(m2,t,i,j);
       if (v!=0) ajoutMatrice3(r,t,i,j,v);
  }
  return r;
}

double minMatrice3(MAT3 m, int t, int nl, int nc) { double val=0;
  LISTE3 p;
  int i;
  for (i=0; i<t; i++)
    for (p=m[i]; !estVideListe3(p); p=p->suiv)
       if (p->val < val) val=p->val;
  return val;
}

MAT3 lectureMatrice3(char* fic, int t,  int* anl, int* anc) { MAT3 r;
  int i,j;
  FILE* fp;
  double val;
  if ( (fp=fopen(fic,"r"))==NULL) return NULL;
  fscanf(fp,"%d %d ",anl,anc);
  if ( (r=allouMatrice3(t)) ==  NULL) return NULL;
  while ( (fscanf(fp,"%d %d %lf ",&i,&j,&val)==3))
      ajoutMatrice3(r,t,i,j,val);
  fclose(fp);
  return  r;
}

void afficheMatrice3(MAT3 m, int t, int nl,int nc) {
  int i,j;
  for (i=0; i<nl; i++) {
    for (j=0; j<nc; j++)
      printf("%5.2lf ",elementMatrice3(m,t,i,j));
    puts("");
  }
}

void afficheTable3(MAT3 m, int t) { int i;
  for (i=0; i<t; i++) {
    afficheListe3(m[i]);
    puts("");
  }
```

```
}
/*
  Taille occupee = 4+ t*4 + k*(4+4+4+8)
  Complecite : O(k/t)
*/

void  main3() { int nl,nc;
  MAT3 m1,m2,m3;
  int t=3;
  m1=lectureMatrice3("data201705_1.dat",t,&nl,&nc);
  afficheTable3(m1,t);
  puts("Matrice m1"); afficheMatrice3(m1,t,nl,nc);
  m2=lectureMatrice3("data201705_2.dat",t,&nl,&nc);
  afficheTable3(m2,t);
  puts("Matrice m2"); afficheMatrice3(m2,t,nl,nc);
  m3=sommeMatrice3(m1,m2,t,nl,nc);
  afficheTable3(m3,t);
  puts("Matrice m3"); afficheMatrice3(m3,t,nl,nc);
  printf("le min est : %lf\n",minMatrice3(m1,t,nl,nc));
}

int main() {
  main1();
  main2();
  main3();
  exit(1);
}

/*
Comparaion  entre S2 et S3 :
S2 - S3 = (t+k-nl)*4
Donc quand on a des matrices de tres grandes tailles (nl grand  >>t)  tres
    creuses (k petit, k<<<nl), solution 3 bin meilleure
*/
```