```c
#include <stdio.h>
#include <stdlib.h>
#include "retroproj.h"
#include <math.h>

#define PI 3.14159265359

void afficherMatice(double** matrice,int Rx, int theta_max){
  for(int i =0;i<Rx;i++){
    for (int j =0; j<theta_max;j++){
      printf("%f\t",matrice[i][j]);
    }
    printf("\n");
  }
  printf("fin affichage\n");
}

double** chargement(char * radon, char * info){
  /*fonction permettant le chargement en memoire sous la forme d'un tableau de la
  transformer de Radon*/
  /*ne pas oublier de liberer la mémoire après les traitements*/
  FILE * fichier_info;
  fichier_info = fopen(info,"r");
  int Rx;
  int theta_max;
  int xp_offset;

  fscanf(fichier_info,"Rx = %d, theta_max = %d, xp_offset = %d\n",&Rx,&theta_max,&xp_offset);
  fclose(fichier_info);

  FILE * fichier_radon;
  fichier_radon = fopen(radon,"r");

  /* chargement de la matrice de la transformer de Radon dans un tableau local*/


  double R[Rx][theta_max];
  for(int i =0;i<Rx;i++){
    for (int j =0; j<theta_max;j++){
      fscanf(fichier_radon, "%lf\t",&R[i][j]);
      //printf("%lf\n", R[i][j]);
    }
    fscanf(fichier_radon,"\n");
  }

  /* creation d'un tableau alloer dynamiquement en meémoir pour pouvoir retourner la
  transformer de Radon*/

  double ** R_tab;
  R_tab = malloc((Rx)*sizeof(double));

  for(int u =0;u<Rx;u++){
    R_tab[u] = malloc((theta_max)*sizeof(double));
    for (int v =0; v<theta_max;v++){
      R_tab[u][v] = R[u][v];
    }
  }
  printf("chargement ok\n");
  return R_tab;
}

int chargementXp_offset(char* file){
  FILE * fichier_info;
  fichier_info = fopen(file,"r");
  int Rx;
  int theta_max;
  int xp_offset;

  fscanf(fichier_info,"Rx = %d, theta_max = %d, xp_offset = %d\n",&Rx,&theta_max,&xp_offset);
  fclose(fichier_info);
  return xp_offset;
```

```
}

void liberationMem(double ** R, int Rx){ /*erreur dans les free (revoir l'allocation de
mémoire, erreur dans l'allocation d'un pointeur ?)*/
  for(int i =0;i<Rx;i++){
    free(R[i]);
  }
  //free(R);
}

double** retroprojectionDiscrete(double** proj, char * info){

//chargement des differentes données
  FILE * fichier_info;
  fichier_info = fopen(info,"r");
  int Rx;
  int theta_max;
  int xp_offset;

  fscanf(fichier_info,"Rx = %d, theta_max = %d, xp_offset = %d\n",&Rx,&theta_max,&xp_offset);
  fclose(fichier_info);

  Rx = 256;
  int k, x, y, u1, u2;
  double u, q;
  double B[Rx][Rx];
  double rad;

  for(k=1;k<theta_max+1;k++){
    rad = (k*PI)/100;
    for(x=0;x<256;x++){
      for(y=0;y<256;y++){
        u=((x-Rx/2)*cos(rad)-(y-Rx/2)*sin(rad)+xp_offset);
        u1 = (int) ceil(u);
        u2 = (int) floor(u);

        if(u1 == u2){
          B[x][y] = B[x][y]+(proj[u1][k])*(PI/theta_max);
        }
        else{
          q = (proj[u2][k]+(u-u2)*((proj[u1][k]-proj[u2][k])/(u1-u2)))*(PI/theta_max);
          B[x][y] = B[x][y] +q;
        }
      }
    }
  }

  printf("fin retroproj\n" );

  double** B_tab;
  B_tab = malloc(Rx*sizeof(double));
  for(int u=0;u<Rx;u++){
    B_tab[u] = malloc(Rx*sizeof(double));
    for(int v=0;v<Rx;v++){
      B_tab[u][v] = B[u][v];
    }
  }
  return B_tab;
}


void ecritMatrice(double ** matrice,int Rx, char * file){

  double m_min = 300.00;
  double m_max =0.00;
  double m;
  int i,j;
  for( i =0;i<Rx;i++){
    for (j =0; j<Rx;j++){
      m = matrice[i][j];
```

```
      if (m_min>m && m>1){
        m_min = m;
      }
      if(m_max < m && m<255){
        m_max = m;
      }
    }
  }
  printf("m_min = %f, m_max = %f\n",m_min,m_max );
  for(i=0;i<Rx;i++){
    for(j=0;j<Rx;j++){
      matrice[i][j] = round((255/(m_max-m_min))*(matrice[i][j]-m_min));
    }
  }


  FILE * fichier_matrice;
  fichier_matrice = fopen(file,"w");

  for( i =0;i<Rx;i++){
    for (j =0; j<Rx;j++){
      fprintf(fichier_matrice, "%lf\t",matrice[i][j]);
    }
    fprintf(fichier_matrice,"\n");
  }
}
```