Describing the role of the functions

Encryption Module

void extrag_dimensiuni(char *,unsigned int *,unsigned int *);
//get the dimensions of the picture

void incarc_poza(char *,pixel **,unsigned int , unsigned int);
//save the picture in an array

void salvez_extern_poza_liniarizata(char *,char *,pixel *,unsigned int, unsigned int);
// save the picture externally

void xorShift(unsigned int,unsigned int **,int , int);
// generate a sequence of 2*W*H random numbers using the seed R0

void get_cheie_secreta(unsigned int *, unsigned int *,char *);
//get the seed and the starting value from a text file

void generare_permutare(unsigned int *, int , int ,unsigned int **);
// generate a permutation using the random sequence R

void permut_pixeli_imagine(pixel *,int , int ,int *);
// function that permutes the pixels of an imagine saved in a linearized matrix

void byte(unsigned int , unsigned char *, unsigned char *, unsigned char *);
// extract the first three bytes from an integer

void criptez_imagine(int *,pixel **,pixel *,unsigned int,unsigned int, unsigned int);
//crypt a given image using the a sequence of random numbers, a permutation and a starting value

void salvez_imagine_criptata(char *,char *,char *,pixel **);
//save the encrypted picture externally

void permutare_inversa(int *, int **,unsigned int,unsigned int);
//generate the inverse permutation

void inversul_criptarii(pixel **,pixel *, unsigned int *,unsigned int ,unsigned int, unsigned int);
//decrypt an encrypted imagine using the inverse permutation and the properties of the xor operation

void imagine_decriptata(char *,char *,char *,pixel *);
//apply the algorithm explained above and save the decrypted image externally

float chi_patrat(int *,float);
// find the value of the chi-squared function for a specific channel

void frecvente_pentru_culoare(char *,pixel *,float *,float *,float *);
// save the value of the chi-squared function for all the three channels

Template Matching Module

void salvez_pixeli_imagine(unsigned int *,char *,unsigned int, unsigned int);**
//save an imagine considering the fact that it is a grayscale one(R=G=B)

float medie_pixel_tablou(unsigned int **, unsigned int, unsigned int);
//compute the average value of a given matrix

double deviatie_standard(unsigned int **,unsigned int, unsigned int);
//calculate the standard deviation of a given matrix

double corelatie(unsigned int **, unsigned int **,unsigned int, unsigned int,double,double);
//compute the correlation between a template S and a window f

void extrag_nume_sabloane(char *,char *);**
// save the names of the templates in an array

void extrag_culori(culoare **, char *);
//save the given colours in an array

void get_fereastra_get_medie_feresatra(int **,int **, unsigned int ,unsigned int,int, int,double *);
//extract a window centered in the point of coordinates (i,j) from the whole imagine

void desenez_contur_fereastra(culoare **,unsigned int, unsigned int,detectii,culoare);
//draw the border of the window with a specific colour C

```
void incarc_imagine_color(char *,culoare ***);
//save the coloured image in a matrix

void salvez_extern_poza(char *,char *,culoare **);
//save an image externally


void gasesc_detectii_sablon_x(int **,culoare **,char *,unsigned int,unsigned int,
unsigned int, unsigned int,float,culoare, detectii *,int *,int);
//find the detections and save them in an array

void template_matching(char *,char *,char **,char *,float,culoare *,detectii **,int *);
//combine the results of the template matching algorithm in an array and save the
result externally



int cmp(const void * , const void *);
//used for the qsort function


void get_colturi_fereastra(coord *,coord *,detectii , unsigned int, unsigned int);
//find the right_top and left_bottom coordinates of a windows centered in (x,y)



double arie_suprapunere_detectii(coord,coord,coord,coord);
//calculate the overlapping area of two given windows



double suprapunere_spatiala(detectii , detectii );
//computer the spatial overlap between two given detections



void eliminare_non_maxime(detectii *, int ,char *,char *,culoare *);
//eliminate those detections which have the spatial overlap over 0.2 and save the
result externally
```