```cpp
/****************************************************************************************
*                                    EXERCISE 2                                         *
*                                   ------------                                        *
*                                  BINARY IMAGES                                        *
****************************************************************************************/

#include <opencv2/opencv.hpp>
#include <iostream>
#include <math>


#define LINES 'L'
#define RECTS 'R'
#define PI 3.14159265358979323846

using namespace std;
using namespace cv;




/****************************************************************************************
* From the pointer to a greyscale image, draws a histogram image and returns a pointer  *
* to it. Two possible styles can be selected:                                           *
* LINES (a line for each bin)                                                           *
* RECTS (a rectangle for each bin)                                                      *
****************************************************************************************/

IplImage * getHistogram(IplImage * gray , char style) {
    int dep = 256;
    int hist[dep];              // Histogram for grayscale image
    int st = 4;                 // To separate lines in the histograms, for better look ;)

    /// Initialize histograms
    for (int i = 0 ; i < dep ; i++)
        hist[i] = 0;

    /// Calculate histogram
    for(int i = 0 ; i < gray->height ; i++) {
        char * ptr = gray->imageData + i * gray->widthStep;
        for(int j = 0 ; j < gray->width ; j++) {
                if( *ptr < 0 ) {
                    char c = *ptr;
                    uchar x = (uchar) c;
                    hist[x] += 1;
                }
                else
                    hist[(unsigned)(*ptr)] += 1;
                ptr++;
        }
    }

    /// Create image for the histograms
    IplImage * his = cvCreateImage(cvSize(st*dep , 600) , IPL_DEPTH_8U , 1);
    cvSet(his , 0);                             // Initialize image (all black)
    his->origin = IPL_ORIGIN_BL;                // Set the origin in the bottom left corner
    /// Draw a line/Rectangle for each bin
    if (style == LINES)
        for (int i = 0 ; i < dep ; i++)
            if (hist[i] != 0)
                cvLine(his , cvPoint(i*st , 0), cvPoint(i*st , hist[i] / 10) , 150 , 1 , 4);
    else if (style == RECTS)
        for (int i = 0 ; i < dep ; i++)
            if (hist[i] != 0)
                cvRectangle(his , cvPoint(i*st , hist[i] / 10),
                            cvPoint((i+1)*st , 0), 150, -1, 4);
    else {
        printf("\nError. No style selected.\n");
        cvReleaseImage(&his);
        return NULL;
    }

    return his;
}




/****************************************************************************************
* Input: a pointer to a GREY SCALE image, a threshold, 3 pointers to int for the        *
* necessary computation to determine the center of mass.                                *
* Forms a binary image, counts the number of pixels set to 0, sums the values of the 2  *
* coordinates for the pixels set to 0 (used later to calculate the center of mass       *
* coordinates).                                                                         *
*                                                                                       *
* PS. It is possible to calculate the coordinates even in this function but it's done   *
*     later to respect the structure of the exercise                                    *
****************************************************************************************/

IplImage * getBinImagePlus(IplImage * img, unsigned t, int * tot, int * xcm, int * ycm) {
    IplImage * I = cvCreateImage(cvGetSize(img) , IPL_DEPTH_8U , 1);
    for(int i = 0 ; i < img->height ; i++) {
        char * ptr = img->imageData + i * img->widthStep;
        char * p = I->imageData + i * I->widthStep;
        for(int j = 0 ; j < img->width ; j++) {
                if ((uchar)(*ptr) >= (uchar)t)
                    *p = 255;
                else {
                    // In this case we are on the object
                    *p = 0;
                    (*tot)++;
                    (*ycm) += i;
                    (*xcm) += j;
                }
            ptr++;
            p++;
        }
    }
    return I;
}
```

```
/****************************************************************************************
* Interactively allows to select a threshold and see the position in a histogram.       *
* When the user is satisfied, it shows the binary image built with the selected         *
* threshold and if the result is not satisfying, allows to select a new threshold again.*
* It returns a pointer to the final binary image that the user confirmed.               *
*
*                                                                                       *
* The function needs:                                                                   *
*    - A pointer to a GREY SCALE image to apply the threshold to.                       *
*    - A pointer to a preliminary histogram image (necessary to restore the histogram   *
*      when changing the threshold).                                                    *
*    - A pointer to a 3-channel image for the histogram to be updated when a new        *
*      threshold is selected (it must be passed as input because the function must      *
*      return another pointer).                                                         *
*    - 3 pointers to int to compute the center of mass coordinates                      *
****************************************************************************************/

#define NEW 'N'
#define OK 'O'

IplImage * interactiveThresholding(IplImage * G, IplImage * Chis, IplImage * his, int *tot, int
                                   *xcm, int *ycm) {
    unsigned int t = 0;                    // Threshold
    char st = 4;                           // To place correctly the threshold line in the image
    IplImage * bin;                        // To contain the binary image

    printf("\nInsert Threshold: ");
    scanf("%u", &t);

    while(1) {
        while(1) {
            if ( !(t >= 0 && t <= 255) ) {
                printf("\nInvalid value!\n");
                printf("\nInsert Threshold: ");
                scanf("%u", &t);
                continue;
            }
            cvCvtColor(his , Chis , COLOR_GRAY2RGB);
            cvFlip(Chis , Chis , 0);
            cvLine(Chis , cvPoint(t*st , 0) , cvPoint(t*st , 600) , cvScalar(0,0,255) , 2 , 4);
            cvShowImage("histogram" , Chis);

            printf("\nPress 'O' to continue or 'N' to select a new threshold (in the histogram
                   image window)\n");

            int c = cvWaitKey(0);

            if (c == OK)
                break;
            else if (c == NEW) {
                printf("\nInsert Threshold: ");
                scanf("%u", &t);
                continue;
            }
            else {
                printf("\nERROR! Press 'O' or 'N' next time.\n");
                printf("\nInsert Threshold: ");
                scanf("%u", &t);
                continue;
            }
        }
        // Note: in this context, tot, xcm, ycm are pointers
        bin = getBinImagePlus(G, t, tot, xcm, ycm);
        cvNamedWindow("Binary Image" , 0);
        cvShowImage("Binary Image" , bin);
        printf("\nPress 'O' if you are satisfied, or 'N' to select a new threshold (in the
               binary image window)\n");

        int k = cvWaitKey(0);

        if (k == OK)
            break;
        else if (k == NEW) {
            printf("\nInsert Threshold: ");
            scanf("%u", &t);
            continue;
        }
        else {
            printf("\nERROR! Press 'O' or 'N' next time.\n");
            printf("\nInsert Threshold: ");
            scanf("%u", &t);
            continue;
        }
    }
    return bin;
}
```

```cpp
/****************************************************************************************
* This function allows the user to select a threshold and draws a new histogram with a  *
* visual indication of the threshold. At the same time computes the necessary numbers   *
* for the centre of mass.                                                               *
*                                                                                       *
* The function needs:                                                                   *
*    - a pointer to a GREY SCALE image to apply the threshold to.                       *
*    - a pointer to a preliminary histogram image (starting point to draw the new       *
*      histogram with the threshold).                                                   *
*    - 3 pointers to int to compute the center of mass coordinates.                     *
****************************************************************************************/

IplImage * simpleThresholding(IplImage * G, IplImage * his, int *tot, int *xcm, int *ycm) {
    unsigned int t = 0;               // Threshold
    char st = 4;                      // To place correctly the threshold line in the image
    IplImage * bin;                   // To contain the binary image

    printf("\nInsert Threshold: ");
    scanf("%u", &t);

    IplImage * Chis = cvCreateImage(cvGetSize(his) , IPL_DEPTH_8U , 3);

    // Represent a grey scale histogram in a 3-channel image.
    // It will be useful later to draw the threshold on the histogram.
    cvCvtColor(his , Chis , COLOR_GRAY2RGB);
    cvFlip(Chis , Chis , 0);

    cvLine(Chis, cvPoint(t*st , 0) , cvPoint(t*st , 600) , cvScalar(0 , 0 , 255), 2, 4);
    cvShowImage("histogram", Chis);
    cvSaveImage("histogram.png", Chis);
    cvReleaseImage(&Chis);

    bin = getBinImagePlus(G, t, tot, xcm, ycm);
    cvNamedWindow("Binary Image", 0);
    cvShowImage("Binary Image", bin);

    return bin;
}
```

```cpp
/****************************************************************************************
* Given a pointer to a binary image and the coordinates of the centre of mass, and 3    *
* pointers to location where the results will be stored, it calculates the reduced      *
* central moments.                                                                      *
* It prints out the central moments as well as the reduced central ones.                *
****************************************************************************************/

void reducedCentralMoments(IplImage * bin, double xcm, double ycm, double * M_20, double * M_02,
                           double * M_11) {
    long tot = 0;
    double b, r;

    for(int i = 0 ; i < bin->height ; i++) {
        char * p = bin->imageData + i * bin->widthStep;
        for(int j = 0 ; j < bin->width ; j++) {
            if ((uchar)(*p) == 0) {
                tot++;
                b = j - xcm;
                r = pow(b, 2);
                *M_20 += r;
                b = i - ycm;
                r = pow(b, 2);
                *M_02 += r;
                *M_11 += (i - ycm) * (j - xcm);
            }
            p++;
        }
    }

    printf("\nCentral Moments:\n\tm_11 = %.2f\n\tm_02 = %.2f\n\tm_20 = %.2f\n", *M_11, *M_02,
           *M_20);

    /// Reduced Central Moments
    *M_02 /= tot;
    *M_20 /= tot;
    *M_11 /= tot;

    printf("\nReduced Central Moments:\n\tm_11 = %.2f\n\tm_02 = %.2f\n\tm_20 = %.2f\n", *M_11,
           *M_02, *M_20);
}
```