```cpp
#include <opencv2/opencv.hpp>
#include <iostream>
#include <string>
#include <stdlib.h>

#define LINES 'L'
#define RECTS 'R'

using namespace std;
using namespace cv;

    /**************************************************************************
    *                              EXERCISE 1                                 *
    *                             -----------                                 *
    *                      IMAGE ACQUISITION AND DISPLAY                      *
    **************************************************************************/

void printing(IplImage *, char []);
void drawHistogram(char [], IplImage *, char);
void drawHistChannels(IplImage *, char);
void drawHistChannelsAlt(IplImage *, char);
void func(uchar, int *);

IplImage * getHistogram(IplImage *, char); // Not used in this exercise

int main(int argc, char *argv[]) {

    /**************************************************************************
    *                                PART 1                                   *
    *                                --------                                 *
    *                        Loading an image from file                       *
    **************************************************************************/

        /// Load image from file specified from command prompt
    IplImage * img;
    img = cvLoadImage(argv[1], 1);

        /// Displays relevant information on command prompt
    printing(img, argv[1]);


    /**************************************************************************
    *                                PART 2                                   *
    *                                --------                                 *
    *                      Image Display & Histogram                          *
    **************************************************************************/

    ///_______________________________Display_____________________________________

    const char * wName = argv[1];
    cvNamedWindow(wName, 0);
    cvShowImage(argv[1], img);

    ///___________________Conversion RGB to GS & Display_____________________

    const char * wGray = "Gray Scale";
    cvNamedWindow(wGray, 1);
    IplImage * gray = cvCreateImage(cvGetSize(img), IPL_DEPTH_8U, 1);
    cvCvtColor(img, gray, CV_RGB2GRAY);
    cvShowImage(wGray, gray);

    ///        Alternatively: Upload directly in GS & Display.
    /// BEWARE: The resultant image has different pixel values AFTER saving it
    /// compared to that obtained from a conversion!
    /*-----------------------------------------------------------------
    const char * wGray1 = "Gray Scale 1";
    cvNamedWindow(wGray1, 0);

    IplImage * gray = cvLoadImage(argv[1], CV_LOAD_IMAGE_GRAYSCALE);
    cvShowImage(wGray, gray);
    //-----------------------------------------------------------------
    //*/

    ///_____________________________Histogram_____________________________________

    /**
    NB: Although not required by PART 2, each channel of the colour image is
    isolated and shown (in gray scale) and a histogram for each is shown.
    It is required in part 3 for a frame captured with a webcam but I decided
    to move it here because many computation can be done in parallel.
    */

        /// Extracting and showing the channels
    /*IplImage * R = cvCreateImage(cvSize(img->width, img->height), IPL_DEPTH_8U, 1);
```

```
    IplImage * G = cvCreateImage(cvSize(img->width, img->height), IPL_DEPTH_8U, 1);
    IplImage * B = cvCreateImage(cvSize(img->width, img->height), IPL_DEPTH_8U, 1);

    cvSplit(img, B, G, R, NULL);

    cvNamedWindow("Red", 0);
    cvNamedWindow("Green", 0);
    cvNamedWindow("Blue", 0);

    cvShowImage("Blue", B);
    cvShowImage("Green", G);
    cvShowImage("Red", R);*/

        /// histograms
    char hName[] = "Histogram";
    drawHistogram(hName, gray, LINES);
    //drawHistChannels(img, LINES);
    //drawHistChannelsAlt(img, LINES);


    /**************************************************************************
    *                               PART 3                                    *
    *                              --------                                   *
    *                     Image acquisition from cameras                      *
    **************************************************************************/

    ///               Display stream from webcam on a window
    ///                   Capture an image from a webcam

    //*************************************************************************

    /*                            DOESN'T WORK ON MY PC
    int c;
    IplImage* color_img;
    CvCapture* cv_cap = cvCaptureFromCAM(-1); // -1 = only one cam or doesn't matter
    cvNamedWindow("Video",0); // create window
    for(;;) {
        color_img = cvQueryFrame(cv_cap); // get frame
        if(color_img != 0)
            cvShowImage("Video", color_img); // show frame
        c = cvWaitKey(10); // wait 10 ms or for key stroke
        if(c == 27)
            break; // if ESC, break and quit
    }
    // clean up
    cvReleaseCapture( &cv_cap );
    cvDestroyWindow("Video");
    //*/

    //*************************************************************************

    /*                     DOESN'T WORK ON MY PC - EMPTY WINDOW
    const char * cName="Hello world!";
    cvNamedWindow(cName, 0);
    CvCapture* capture = 0;
    double capProp=0;
    IplImage *frame, *frame_copy=0;
    capture = cvCaptureFromCAM(1);
    if (capture) {
        for (;;) {
            if ( !cvGrabFrame( capture ))
                break;
            frame = cvRetrieveFrame( capture );
            if ( !frame )
                break;
            if ( !frame_copy ) {
                printf("\nFrame settings:\n Width: %d\n Height: %d\n",
                frame->width, frame->height);
                frame_copy = cvCreateImage (cvSize(frame->width,frame->height),
IPL_DEPTH_8U, frame->nChannels);
                cvResizeWindow(wName,frame->width,frame->height);
            }
            if ( frame->origin == IPL_ORIGIN_TL )
                cvCopy ( frame, frame_copy, 0 );
            else
                cvFlip ( frame, frame_copy, 0);
            cvShowImage(wName, frame_copy);
            cvWaitKey(30);
            //if (cvWaitKey(5)>0)
               // break;
        }
    }
    //*/

```

```
    //*****************************************************************************
    /*              WORKS IN THE LAB, DOESN'T WORK ON MY PC - CRASHES OR
    //             TAKES AN EMPTY FRAME BUT ONLY IF THE CAMERA IS TURNED ON...

    IplImage * frame;
    CvCapture * capture = 0;
        // initialize capture device
    //capture = cvCaptureFromCAM(0);
    capture = cvCreateCameraCapture(-1);
    if(!cvGrabFrame(capture))
        printf("error");
    frame = cvRetrieveFrame(capture);

    printf("\nFrame settings:\n Width: %d\n Height: %d\n", frame->width, frame->height);

    cvNamedWindow("Cam", 2);
    cvResizeWindow("Cam", frame->width, frame->height);
    cvShowImage("Cam", frame);
    cvWaitKey(30);
    cvSaveImage("frame.png", frame);

    //*/

    //*****************************************************************************
    //             STREAM FROM WEBCAM - WORKS ONLY IF THE CAMERA IS OFF
    //                     IF NOT, SHOWS A BLACK WINDOW...
    //                           ...AND IT'S C++
    /*
    VideoCapture cap(1);

    /// cap is an object and to control the properties of the camera the methods set
    /// and get can be used:
    ///                       cap.set(...);
    ///                       cap.get(...);

    Mat frame;
    if (!cap.isOpened()) {
        printf("Cam could not be accessed\n");
        return -1;
    }
    namedWindow("Cam");
    while(cap.read(frame)) {
        imshow("Cam", frame);
        if (waitKey(10) >= 0) {
            IplImage * im_cam = new IplImage(frame);
            cvSaveImage("Cam.png", im_cam);
            break;
        }
        if(frame.empty()) {
            printf("End of stream\n");
            break;
        }
    }

    */
    //*****************************************************************************
    /*       CAPTURE AN IMAGE FROM WEBCAM - WORKS ONLY IF THE CAMERA IS OFF.
    //                       IF NOT, CAPTURES A BLACK IMAGE...
    //                              ...AND IT'S C++

    VideoCapture cap(1);
    Mat frame;
    if (!cap.isOpened()) {
        cout <<"Cam could not be opened"<<endl;
        return -1;
    }
    namedWindow("Cam");
    cap.read(frame);
    imshow("Cam", frame);
    IplImage * im_cam = new IplImage(frame);
    cvNamedWindow("CamIPL",0);
    cvShowImage("CamIPL", im_cam);
    //*/

    //*****************************************************************************

    /*********************************************************************
    *                               PART 4                               *
    *                               ------                               *
    *                          Image storage                             *
    *********************************************************************/
    /*cvSaveImage("Grey Scale.png", gray);
    cvSaveImage("Blue.png", B);
    cvSaveImage("Red.png", R);
```

```
    cvSaveImage("Green.png", G);
    //cvSaveImage("Cam.png", im_cam);
    //imwrite("frame.png", frame); // If you're using the C++ solution without
                                   // without converting to IplImage.

    */
    ///_____________________________Clean-up_____________________________
    cvWaitKey(0);

    cvReleaseImage(&img);
    cvReleaseImage(&gray);
    //cvReleaseImage(&R);
    //cvReleaseImage(&G);
    //cvReleaseImage(&B);
    //cvReleaseImage(&im_cam);
    //cvReleaseImage(&frameCopy);
    //cvReleaseImage(&frameC);

    cvDestroyAllWindows();

    return 0;
}



    /*****************************************************************************
    * Simply shows in the command prompt relevant information about the uploaded *
    * image.                                                                     *
    *****************************************************************************/
void printing(IplImage * img, char name[]) {
        /// File Name
    printf("File Name: %s\n", name);

        /// File Size
    long imgSize = img->imageSize;    // in Bytes, before compression
    imgSize = imgSize / 1000;         // conversion to KB
    printf("File Size (no compression): %ld KB\n", imgSize);
    /** PROBLEM: The size stored in iplImage structure doesn't take into account
    the compression to get a final jpg image so the size obtained is different.
    I can't find any solution that actually works...*/

        /// Image Format - Stupid but works!
    char * ext = strrchr(name, '.');
    if (!ext)
        printf("No extension found\n");
    else
        printf("Image format: %s\n", ext + 1);

        /// Image Size
    printf("Image Size: %d x %d\n", img->width, img->height);

        /// Color Format - Not very nice...
    if (img->nChannels == 3)
        printf("Colour format: Coloured\n");
    else if (img->nChannels == 1)
        printf("Colour format: Gray Scale\n");
    else
        printf("Colour format: Unknown\n");
}


    /*****************************************************************************
    * Takes a pointer to a GREY SCALE image, a style identifier, and a string.   *
    * The function draws the histogram by drawings lines or rectangles for each  *
    * bin (as specified by 'style') then saves the histogram in an image named   *
    * as specified by the string and also display it in a window whose title is  *
    * specified by the string as well.                                           *
    *                                                                            *
    * NB. Having to do everything by itself, there is some clumsy processing to  *
    * manage the name of the file and the title of the window.                   *
    *****************************************************************************/
void drawHistogram(char name[], IplImage * gray, char style) {
    int dep = 256;
    int hist[dep];   // Histogram for gray scale 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++) {
```

```
                func((uchar)(*ptr), hist);
                ptr++;
            }
        }

        /// Create image for the histogram
    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

    cvNamedWindow(name, 2);         // Create window to contain the image

        /// Draw the histogram - 2 different styles
    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;
    }

        /// Save Image in png format (next 4 lines to add the extension)
    int l = strlen(name);
    char filename[l + 4];
    strcpy(filename, name);
    strcat(filename, ".png");

    cvSaveImage(filename, his);

    cvShowImage(name, his);

    cvReleaseImage(&his);
}


    /**************************************************************************
    * Just a support function to 'drawHistogram', 'drawHistChannels', and     *
    * 'getHistogram'.                                                         *
    **************************************************************************/
void func(uchar c, int * h) {
    unsigned x = c;
    (*(h + x))++;
}


    /**************************************************************************
    * From a BGR image extracts the three channel (grey scale images) and draws  *
    * and saves the images and relative histograms.                           *
    * Two possible styles to draw the histograms: LINES or RECTS              *
    **************************************************************************/
void drawHistChannels(IplImage * img, char style) {
    IplImage * R = cvCreateImage(cvSize(img->width, img->height), IPL_DEPTH_8U, 1);
    IplImage * G = cvCreateImage(cvSize(img->width, img->height), IPL_DEPTH_8U, 1);
    IplImage * B = cvCreateImage(cvSize(img->width, img->height), IPL_DEPTH_8U, 1);
    cvSplit(img, B, G, R, NULL);
    int dep = 256;
    int histB[256], histG[256], histR[256];   // Histograms
    int st = 4;                               // To separate lines in the histograms

        /// Initialize histograms
    for (int i = 0; i < dep; i++) {
        histB[i] = 0;
        histG[i] = 0;
        histR[i] = 0;
    }

        /// Calculate histogram
    for(int i = 0; i < img->height; i++) {
        char * ptrB = B->imageData + i * B->widthStep;
        char * ptrG = G->imageData + i * G->widthStep;
        char * ptrR = R->imageData + i * R->widthStep;
        for(int j = 0; j < img->width; j++) {
                func((uchar)(*ptrB), histB);
                ptrB++;

                func((uchar)(*ptrG), histG);
                ptrG++;
```

```
                func((uchar)(*ptrR), histR);
                ptrR++;
            }
        }

        /// Create image for the histograms
        IplImage* hisB = cvCreateImage(cvSize(st * 256, 600), IPL_DEPTH_8U, 1);
        IplImage* hisG = cvCreateImage(cvSize(st * 256, 600), IPL_DEPTH_8U, 1);
        IplImage* hisR = cvCreateImage(cvSize(st * 256, 600), IPL_DEPTH_8U, 1);

        /// Initialize image (all black)
        cvSet(hisB, 0);
        cvSet(hisG, 0);
        cvSet(hisR, 0);

        /// Create window to contain the image
        cvNamedWindow("Blue channel", 2);
        cvNamedWindow("Green channel", 2);
        cvNamedWindow("Red channel", 2);

        /// Set the origin in the bottom left corner
        hisB->origin = IPL_ORIGIN_BL;
        hisG->origin = IPL_ORIGIN_BL;
        hisR->origin = IPL_ORIGIN_BL;

        /// Draw histograms - 2 different styles
        if (style == LINES)
            for (int i = 0; i < dep; i++) {
                if (histB[i] != 0)
                    cvLine(hisB, cvPoint(i*st, 0), cvPoint(i*st, histB[i] / 10), 150, 1, 4);
                if (histG[i] != 0)
                    cvLine(hisG, cvPoint(i*st, 0), cvPoint(i*st, histG[i] / 10), 150, 1, 4);
                if (histR[i] != 0)
                    cvLine(hisR, cvPoint(i*st, 0), cvPoint(i*st, histR[i] / 10), 150, 1, 4);
            }
        else if (style == RECTS)
            for (int i = 0; i < dep; i++) {
                if (histB[i] != 0)
                    cvRectangle(hisB, cvPoint(i*st,histB[i]/10), cvPoint((i+1)*st,0), 150,
-1, 4);
                if (histG[i] != 0)
                    cvRectangle(hisG, cvPoint(i*st,histG[i]/10), cvPoint((i+1)*st,0), 150,
-1, 4);
                if (histR[i] != 0)
                    cvRectangle(hisR, cvPoint(i*st,histR[i]/10), cvPoint((i+1)*st,0), 150,
-1, 4);
            }
        else {
            printf("\nError. No style selected.\n");
            cvReleaseImage(&hisB);
            cvReleaseImage(&hisG);
            cvReleaseImage(&hisR);
            return;
        }

        /// Show on window and save on disk
        cvShowImage("Blue Hist", hisB);
        cvShowImage("Green Hist", hisG);
        cvShowImage("Red Hist", hisR);
        cvSaveImage("Blue Channel Histogram.png", hisB);
        cvSaveImage("Green Channel Histogram.png", hisG);
        cvSaveImage("Red Channel Histogram.png", hisR);

        cvReleaseImage(&B);
        cvReleaseImage(&G);
        cvReleaseImage(&R);
        cvReleaseImage(&hisB);
        cvReleaseImage(&hisB);
        cvReleaseImage(&hisB);
}


    /*****************************************************************************
    * Alternative function to extract the channels of a BGR image and to draw    *
    * the histograms.                                                            *
    * This version uses another function to deal with each single channel one at *
    * a time.                                                                    *
    *                                                                            *
    * PROS: it's easier to understand, maintain, there is not much repeated code *
    *       there are no issues with names, exploits functional organization.    *
    * CONS: for each channel it has to scan an entire image, so 3 images of the  *
    *       size must be scanned compared to 'drawHistChannels' which computes   *
    *       the 3 histograms in parallel... It could be nice if one decides to   *
```

```
    *       treat only a subset of channels, but at present state there is not   *
    *       such flexibility.                                                    *
    ***************************************************************************/
void drawHistChannelsAlt(IplImage * img, char style) {
    IplImage * R = cvCreateImage(cvSize(img->width, img->height), IPL_DEPTH_8U, 1);
    IplImage * G = cvCreateImage(cvSize(img->width, img->height), IPL_DEPTH_8U, 1);
    IplImage * B = cvCreateImage(cvSize(img->width, img->height), IPL_DEPTH_8U, 1);
    IplImage * hisR, * hisG, * hisB;

    cvSplit(img, B, G, R, NULL);

    hisB = getHistogram(B, style);
    hisG = getHistogram(G, style);
    hisR = getHistogram(R, style);

    cvNamedWindow("Blue Hist", 2);
    cvNamedWindow("Green Hist", 2);
    cvNamedWindow("Red Hist", 2);

    cvShowImage("Blue Hist", hisB);
    cvShowImage("Green Hist", hisG);
    cvShowImage("Red Hist", hisR);

    cvSaveImage("Blue Channel Histogram.png", hisB);
    cvSaveImage("Green Channel Histogram.png", hisG);
    cvSaveImage("Red Channel Histogram.png", hisR);

    cvReleaseImage(&B);
    cvReleaseImage(&G);
    cvReleaseImage(&R);
    cvReleaseImage(&hisB);
    cvReleaseImage(&hisB);
    cvReleaseImage(&hisB);;
}


    /*****************************************************************************
    * Forms the histogram for a GREY SCALE image and returns a pointer to the    *
    * histogram image.                                                           *
    * The calling function has to deal with visualization and storing. This      *
    * function does not deal with these aspects.                                 *
    *****************************************************************************/
IplImage * getHistogram(IplImage * gray, char style) {
    int dep = 256;
    int hist[dep]; // Histogram for gray scale 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++) {
                func((uchar)(*ptr), hist);
                ptr++;
        }
    }

        /// Create image for the histograms
    IplImage * his = cvCreateImage(cvSize(st*dep,600), IPL_DEPTH_8U, 1);
    cvSet(his, 0);    // Initialize image (all black)

        /// Create window to contain the image
    //cvNamedWindow("Histogram", 2);

        /// Set the origin in the bottom left corner
    his->origin = IPL_ORIGIN_BL;

        /// 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;
```

```
583     }
584 
585         /// Show on window
586 
587     //cvShowImage(name, his);
588     //name = strcat(name, ".png");
589     //cvSaveImage(name, his);
590     //cvReleaseImage(&his);
591     return his;
592 }
593 
```