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

#include <opencv2/opencv.hpp>
#include <iostream>
#include <string>
#include <stdlib.h>
#include "Ex1CV-header.h"

using namespace std;
using namespace cv;

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
    infoDisplay(img , argv[1]);


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

    ///_______________________________Display________________________________________

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


    ///__________________Conversion RGB to Grayscale & Display________________________

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





    // Alternatively: Upload directly in GS & Display.

    // BEWARE: The result image has different pixel values AFTER saving it
    // compared to that obtained from a conversion!

    /*
    const char * wGray1 = "Gray Scale 1";
    cvNamedWindow(windowGray1 , 0);
    IplImage * gray = cvLoadImage(argv[1] , CV_LOAD_IMAGE_GRAYSCALE);
    cvShowImage(windowGray , gray);
    */

```

```

        ///_________________________________Histogram______________________________________

        /**
        NB: Although not required by PART 2, each channel of the colour image is isolated and
        shown (in grayscale) 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 histogramName[] = "Histogram";
        drawHistogram(histogramName , 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);
    */
```

```cpp
        //*********************************************************************************

        // STREAM FROM WEBCAM - WORKS ONLY IF THE CAMERA IS OFF; IF NOT, IT SHOWS A BLACK
        // WINDOW
        // PS. 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.
        // Images are treated as objects of the class 'Mat' in the C++ version of OpenCV.

        Mat frame;
        if ( !cap.isOpened() ) {
            std::cout << "Cam could not be accessed" << std::endl;
            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() ) {
                std::cout << "End of stream" << std::endl;
                break;
            }
        }


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

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

        VideoCapture cap(1);
        Mat frameCaptured;
        if ( !cap.isOpened() ) {
            std::cout << "Cam could not be accessed" << std::endl;
            return -1;
        }
        namedWindow("Cam");
        cap.read(frameCaptured);
        imshow("Cam" , frameCaptured);
        IplImage * im_cam_captured = new IplImage(frameCaptured);
        cvNamedWindow("CamIPL" , 0);
        cvShowImage("CamIPL" , im_cam_captured);


        //*********************************************************************************
        /*********************************************************************************
        *                                    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_capture);
        //imwrite("frame.png" , frameCaptured);      // If you're using the C++ solution
                                                     // without converting to IplImage.
        */


        ///________________________________Clean-up______________________________________


        cvWaitKey(0);

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

        cvDestroyAllWindows();

        return 0;
    }
```