Eric Miao
47163991

PART A: Exploring OpenCV C++ Basics (32 points)

Instructions: Write a code segment to do the following in OpenCV using the C++ API. Use your OpenCV display environment to write code to verify with a display when appropriate. **Write** the answer here and upload to Canvas.

1) Create a 1024x1024 8-bit 4 channel image, with each element in the range 0..255 ?

```
Mat m1 = Mat(1024, 1024, CV_8UC4);
   namedWindow("M1");
   moveWindow("M1",10,50);
   imshow("M1", m1);
   waitKey(0);
```

2) Create a 600 x 300 solid yellow image?

```
Mat m1 = Mat(600, 300, CV_8UC4, Scalar(0,255,255));
   namedWindow("M1");
   moveWindow("M1",10,50);
   imshow("M1", m1);
   waitKey(0);
```

3) Create a 256x256 image filled with random 32-bit floats in the range [0,1)?

```
Mat m1(256, 256, CV32FC1, Scalar(0));
randu(m1, Scalar(0), Scalar(1));
   namedWindow("M1");
   moveWindow("M1",10,50);
   imshow("M1", m1);
   waitKey(0);
```

4) Create a 9x9 identity matrix.

```
Mat m1 = Mat::eye (9, 9, CV_32F);
namedWindow("M1");
moveWindow("M1",10,50);
imshow("M1", m1);
waitKey(0);
```

5) Create a 20 x 50 matrix initialized to all ones.

```
Mat m1 (20, 50, CV_32F, 1);
namedWindow("M1");
moveWindow("M1",10,50);
imshow("M1", m1);
waitKey(0);
```

6) Read a color image (call it "color.png") convert to gray scale and display.

```
Mat m1 = imread("color.png");
Mat grayImage;
cvtColor(m1, grayImage, COLOR_BGR2GRAY);
namedWindow("GrayImage");
moveWindow("GrayImage",10,50);
imshow("GrayImage", grayImage);
waitKey(0);
```

7) Given a Mat object, m, that has type CV_32FC1 and has 256 rows and 256 columns, write a code segment to find the maximum value.

```
m1.convertTo(m2, int CV_32FC1); // m1 is our src mat object
float max = 0.0;
for (int i = 0; i < m.rows; i++){
    for(int j = 0; j < m.cols; j++){
        if (m.at<float>(i,j) > max){
            max = m.at<float>(i,j);
        }
    }
}
```

8) Given a Mat object, m, that has type CV_8UC3, write a code segment that would change the values of the 64th row to all white. (What is max intensity. Assume there are >= 64 rows in the image.) Apply this to a color image and UPLOAD a SCREEN SHOT.

```
Mat m1 = imread("flowergray.png");
    Mat m2(400, 600, CV_8UC3);
    m1.convertTo(m2, CV_8UC3);
    for(int i = 0; i < m2.cols; i++){
        cv::Vec4b & pixel = m2.at<cv::Vec4b>(64, i);
        pixel[0] = 255;
        pixel[1] = 255;
        pixel[2] = 255;
    }
```

```
namedWindow("m2");
    moveWindow("m2",10,50);
    imshow("m2", m2);
    waitKey(0);
```

PART B: Generating Images using OpenCV C++ (30 points)

Instructions: Submit each of these to canvas as single source code files for each problem for a completion grade. Include a brief header. This is "exploration code" not "delivery code" so style and error handling should be for yourself. Be prepared to demo and explain via zoom share screen when asked - so keep your code organized.

1) Write an program, **ChannelDisplay.cpp**, that will produce a 4-window display image with the original image displayed in one and each color channel displayed in the others.

2) Create an interactive program, **ThresholdSlider.cpp** , with two sliders that control two threshold values t1 and t2, where 0 <= t1 <= t2 <= 255. Apply these values to a CV_8UC1 grayscale image, I, to produce a new image IT
Display both the original window and the threshold output.

3) Create a program, **AddRandomNoise.cpp**, to generate and display a new 256x256 gray scale image that adds random noise to flowersgray.tiff. HINT: Use the matrix you generated from Part A, problem 3

<u>PART C: MiniProject (38 points)</u>
Instructions:

Develop an interactive tool for creative 3-channel photo editing, suitable for demonstrating and sharing with the class.

Upload a single source code file to Canvas, provide useful comments. use good coding style and add reasonable error handling. Your header must include a brief usage description.

Also upload a representative screenshot of your working program or output to a specific image.

MINIMUM: Create a portrait creation tool that applies any sort of thresholding to three separate color channels, with a slider control for each channel.

MORE FUN: Experiment. You may rearrange the original color channels and you may choose to use a single threshold or a range of thresholds. Or otherwise mess around with data, columns, rows.

<u>Points distributions</u>

Compiles and runs (10 points)
Coding style – (10 points)
Functional Requirements – performs as required (18 points)

Up to 5 bonus points up for grabs for exception work, which can be technical merit, creative interest, or exemplary presentation (demo or usability)