**General Requirements:**

1. Please place semicolons to avoid unnecessary console output, such as:

I=imread(‘1.jpg’);

2. In your main function, place a message “----Done for Question \*----” followed by a pause command (i.e., wait for a key to be pressed before continuing) at the end of each solution, where \* is the question number (1, 2, 3, …), such as:

*disp('-----Done for Question 2-----');*

*pause;*

3. Please submit your zipped .m files via the Blackboard system. The zipped file only has .m files without any image files.

**Grading:**

Q1-Q4: 15 points for each

Q5: 40 points

**Questions:**

1. Load the image ***peppers.bmp*** into a variable ***A***.

Display the loaded image ***A*** on figure 1 with the figure title as “Original RGB color Image”.

2. Convert image ***A*** into a grayscale image and store it as ***B***.

Transpose image ***B*** as ***TB***.

Vertically flip image ***B*** as ***VB***.

Horizontal flip image ***B*** as ***HB***.

See *flipud* and *fliplr* functions in Matlab.

Display images ***B***, ***TB***, ***VB,*** and ***HB*** on figure 2 with the corresponding location of upper left, upper right, bottom left, bottom right with corresponding title “***B”***, “***TB”***, “***VB”,*** and “***HB”.***

Display the maximum, minimum, mean, and median intensity value of ***B*** on the Matlab console.

3. Normalize image ***B*** to ***C***, whose data type is **double** and whose values fall in the range of [0, 1]. Display image ***C*** on figure 3 with the title “Normalized Grayscale Image” as the figure title.

Save image ***C*** in the png format to a file called “\*\_C.png” where \* should be your first name.

4. Perform binary thresholding on the original normalized grayscale image ***C***. A threshold 0.4 is chosen and all values in ***C*** greater than the threshold are set to 1, otherwise set to 0, leading to a new image ***bw1***. Please finish this task without calling the Matlab built-in function *im2bw.* Then, please finish the same task by calling the Matlab built-in function *im2bw,* leading to a new image ***bw2.***

Display ***bw1*** and ***bw2*** side-by-side on figure 4 and label the two images with “My method bw1” , and “Matlab method bw2”, respectively.

If ***bw1*** and ***bw2*** are the same, display “bw1 and bw2 are the same” as the console output, otherwise display “bw1 and bw2 are not the same” as the console output.

5. Write a Matlab function **MyBlur** to replace all 16 pixels in each non-overlapping 4×4 block of any input image with their average intensity value. The prototype of **MyBlur** function is defined as follows:

**function [BI] = MyBlur(I);** where I is the original image and BI is the blurred image, where I can be either one 3-channel RGB color image or one single-channel grayscale image.

See *mean* function in Matlab.

Call this function in your main script to blur the color image ***A*** and get the blurred image to variable ***A2***.

Call this function in your main script to blur the grayscale image ***B*** and get the blurred image to variable ***B2***.

Display images ***A***, ***B***, ***A2*** and ***B2*** from left to right and top to bottom with the appropriate titles on figure 5.