Due Date:
Sunday, September 11 at 11pm (submit to blackboard)

General Instructions:
Each problem should be a separate section of your MA2 script submitted through blackboard. All questions should have appropriate commenting and be easy to follow. Test cases and sample output are provided for each problem; however, your code should work for any case.

Part I: Matrix Manipulation

Problem #1: Chapter 16: ICA 16-4 – Follow the instructions provided in the question from your textbook. Make sure to comment out code that does not work and provide a comment explaining what the error would be. Formatted output is not required for the problem.

Problem #2: Chapter 17: ICA 17-4 – Follow the instructions provided in the question from your textbook. Make sure to comment out code that does not work and provide a comment explaining what the error would be. Formatted output is not required for the problem.

Problem #3: Chapter 16: Review Question 1. Follow the instructions provided in the question from your textbook. Formatted output is not required for the problem. The grade will look at your Workspace to determine if the correct answers were determined.

PART II: Image Processing (Pokémon Go)

Background: Types of Image Processing in MATLAB

RGB image: This is another format for color images. It represents an image with three matrices of sizes matching the image format. Each matrix corresponds to one of the colors red, green or blue and gives an instruction of how much of each of these colors a certain pixel should use.

When working with real colored images your array will be a 3-Dimensional matrix indexed by [x y z], where the [x,y] indexes the spatial dimensions of the image, while the z value (1,2, or 3) indexes the RGB (red, green, and blue) color components respectively. Every image is a combination of the three colors and can be broken down into each individual color as seen in Figure 1 below.

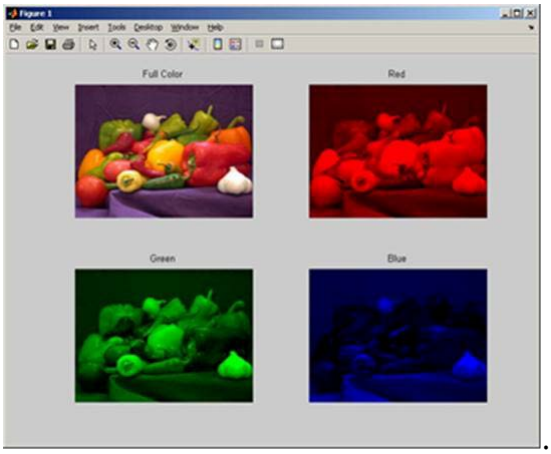

Figure 1: Full colored image broken down into its RGB image components

Binary image: This image format also stores an image as a matrix but can only color a pixel black or white (and nothing in between). It assigns a 0 for black and a 1 for white. See Figure 2 below

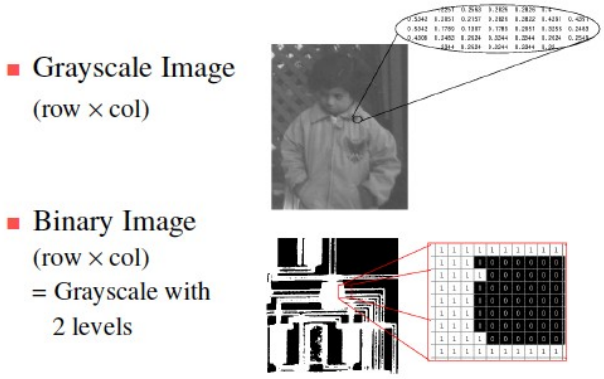

Figure 2: Grayscale image versus binary image showing assigned values

Indexed image: This is a practical way of representing color images. (In this course, we will mostly work with gray scale images, but once you have learned how to work with a gray scale image, you will also know the principle how to work with color images.) An indexed image stores an image as two matrices. The first matrix has the same size as the image and one number for each pixel. The second matrix is called the color map and its size may be different from the image. The numbers in the first matrix is an instruction of what number to use in the color map matrix.

Image Manipulation: Since all images are matrices, it naturally follows that executing simple matrix operations on an image matrix will alter an image. In this project, we will explore various ways to alter an image by running matrix operations.

By the end of the semester, you will have accumulated several methods to edit a single photo. By the end of this assignment, you will have completed a series of commands that are used by image editing programs. These are presets that you often find in Photoshop, or even your cellphone photo-editing App. Hopefully, at the end you can prompt the user to edit the photo in several ways. Finally, you will find that you created your very own image-editing program

Problem #4a: Altering an image

Task 1: Load in the image of your choice named "background.jpeg", and assign it to a variable name ***BACKGROUND***. Make sure the files is RGB file (jpeg) and is a size greater than____. Display the background matrix as an image and label Figure 1.

Task 2: Ask the user to input a number between 0 and 255. Assign this number to a variable named ***UserSubtract***. Subtract that number to every entry in the image matrix from Task 1 and store the new matrix as ***BACKGROUND_Sub***. Display the updated picture by creating a new figure and label it Figure 3.

Task 3: Ask the user to input a number between 0 and 255. Assign this number to a variable named ***UserAdd***. Add that number to every entry in the image matrix from Task 1 and store the new matrix as ***BACKGROUND_Add***.

Task 4: Ask the user to input a number between 0 and 3 (decimals accepted) and assign it to a variable named, ***Multiplier***. Multiply the image matrix from Task 1 by the user value and store the new matrix as ***BACKGROUND_Mult.***

Task 5: Create a red tinted image of your background by removing the green and blue color from the image. Store the new matrix as ***BACKGROUND_tint.***

Task 6: Display all four matrices as images on one plot (BACKGROUND_Sub, BACKGROUND_Add, BACKGROUND_Mult, BACKGROUND_tint) as shown below. HINT: You will need to use subplot.

Task 7: As comments in your code, describe how each matrix math changed the image. Do not display on the command window.

Problem #4b: Superimposing Pokemon on the Background

Task 1: You are provided 4 different Pokemon characters on a green screen. Load each of the Pokemon images and assign the variable name associate with their character (filename is its character name).

Task 2: Display in a 2 x 2 subplot the 4 Pokemon characters on the green screens.

Task 3: Using the masking technique described in class and in the extra resources on blackboard, identify the ALL GREEN CELLS in each pokemon image and replace it with the values in from your ***BACKGROUND***. Do this for each Pokemon character and save a new matrix for character and background as ***CharacterName_Back***.

Task 4: Display in a 2 x 2 subplot the 4 Pokemon characters on your background.