Homework #1.

Instructions: Please save your final code as a single *.m file with the format **LASTNAME_FIRSTNAME_homework1.m and upload to Canvas.** To put comments in this m-file, put a '%' in front of your text. For example, a note or comment would look like this:

> *% this would be a comment, and if you copied and pasted this line into the*
> *% command window, it wouldn't execute. For a clean division*
> *% between question numbers, please use '%%', which creates a section break.*

Although some of these problems should be easy and relatively straightforward, others may be more difficult and take some work and creative thinking. Keep in mind that you've learned the tools and functions that are required to solve them (or, you should be able to figure out the needed functions, in the case of #6 and #7 below). If you get completely stuck on some of these, feel free to get ideas/advice/approaches from a classmate, friend, or ask me, but ultimately the work you submit should be your own. Even if you can't get the right answer, you should still submit code with some of what you've tried (better than leaving it blank) so I can see how you attempted to solve it. Keep in mind that however difficult it seems, you're still learning from trying, so that's good. Also, if you're stuck for a while, try taking a break and then coming back to it later.

%% Problems

1. Load in a *.jpg file of your choice. Place a blue square so that it covers up the center 50 x 50 pixels of the image. Make sure you write your code flexibly enough so that you or I can run this same code using another image, without having to change anything in your code except the image file name (i.e., no calculations by hand, use variables with values based on "size" as opposed to specifying numbers (e.g., 640 wide x 400 long), etc.)

2. Downsample the resolution of your image by 2. (e.g., if it's originally 1000 x 500 pixels, make it 500 x 250). Again, make your code flexible so you can run it on an image of any size. (note: make sure it can properly handle pictures with odd numbers).

3. Convert any color image to grayscale, using the following formula (not *rgb2gray* which we learned in class): gray_image = 0.2989*red + 0.5870*green + 0.1140*blue [NOTE: To visualize it with imagesc, you will have to also use the command "colormap gray" as mentioned in class. Otherwise, you get weird colors because it's using "colormap jet" color scheme. See 'help colormap' to learn more about this]

4. Write code that does the following: Using a grayscale image, determine if the mean luminance is less than 128, and if it is, increase the luminance by 20% and display the image on screen. If the mean luminance is greater than 128, decrease the luminance by 20% and display the image on screen.

5. Create a 10 x 1 vector of random integers, v1. Sort the column in ascending order. Then, apply that same sort order to another 10 x 1 vector of random integers, v2.

6. Create a 100 x 2 matrix (M1) of normally distributed random values (mean = 0; std = 1); use an independent sample t-test to compare whether each column come from the same underlying distribution or not. [HINT: I know we haven't gone over how to run t-tests yet, but you should be able to use matlab help to find out in about 30 seconds or less]. There is a lot of output that you can capture from this t-test function. Try capturing/outputting ***only*** two of these -- 'p' and 'stats'.

7. Using this same matrix, M1, visualize all 200 values in a single histogram. Using the 'axis' function, make the x-axis extend from -1.5 to 1.5. Then, replace all positive values with twice their current value. View this new histogram, using an x-axis of -2 to 6.

8. Determine the **percent** of outliers (defined as mean +-3*standard deviation) in the flattened matrix M1; then, covert all of them to NaNs.

9/10. Imagine you have collected data for an experiment where you had 15 subjects complete an experiment that includes 20 trials per day and they complete 5 days of testing each. Using random data, structure this information into a single matrix (you can just use rand or randn to create fake data). Then, do the following:

— Calculate the mean across all trials (i.e., collapse day and trial) for each subject.
— Calculate the mean values of the 4th day for the 7th participant.