**Assignment 3 (60/25 points)**

**Histogram Equalization** ***(20/10)***

1. Read and display an image of your choice. (To help you with the next problem, look for an image that has interesting objects of different intensities.)
2. Calculate and display the histogram of this image.
3. Enhance the contrast of the intensity image using histogram equalization and display both the uniform histogram and the newly enhanced intensity image.
4. Explain why the two histograms (of the original image and of the enhanced image) are different.

**481 Students:** ***(10/0)*** Apply a local enhancement approach on this image and show your results. Before you start, consider how your image might call for a particular window size. For fun, you might want to try a few different window sizes. One student actually put the window size in a loop from 1 to the image size and showed the results in a video. The gauntlet has been thrown. ☺

**Filtering: (*25/15*)**

1. **481 Students: (*10/0)*** write a function to perform filtering. Do not use any of the prewritten functions; your function should process the image pixel by pixel and explicitly calculate the sum of products for each pixel in the image. You can use whatever simplifying assumptions you want for pixels on the border of the image. State those assumptions.
2. **All Students:** perform filtering on the image of your choice using a Prewitt filter, a Sobel filter, a point filter, and the blurring filter. Use built-in functions to do the filtering. **481 students:** use both your function and a library function so that you can compare your results to the library results.

**Bit Plane Splicing *(15/0)***

Bit place splicing (<https://en.wikipedia.org/wiki/Bit_plane>) is a simple form of frequency analysis in which the frequencies are defined by the bits representing the intensity of the pixels. Write a program to perform bit-place splicing on an image such that you can generate a figure similar to the one shown in the Wikipedia article: your original image and each of the 8 bit planes in it. Perhaps the key lesson is that each bit-position represents a different binary image.

Then, “assemble” the original image by successively adding bit planes to the most significant bit plane. You will have 7 new images, which will be the combination of bit planes 7 and 6; 7 and 6 and 5; 7 and 6 and 5 and 4; … all bit planes, which should be the original image. State which bit plane you feel you could stop at and still get a good visual match with the original image.

**General submission instructions:**

1. Be kind to your aging, over-worked professor and submit only a single document. This can be pdf, MS Word, OpenOffice, etc. Do not submit a zip file.
2. Your single document should include the input image for your problem, if required, and answers to each of the sub-problems (text, image or both, as appropriate). Your document should also include code that you wrote to generate your answers.
3. You may use any images you like for the programming; I encourage you to use images that might be useful/interesting for your final project.
4. Feel free to use whatever functions MatLab supplies. Also feel free to write your own, if you are so inclined; it will take more time, but you will gain a deeper understanding of the material.
5. Point values for each question are indicated as **(*x/y)*** in which ***x*** is the point value for 481 students and ***y*** is the point value for 381 students.