1. Your submission should include **a very very clear README describing how to run it**.

2. **Report will be the first thing that we will look at from your submission.**

Notes from instructor:

- Use MATLAB or Python for your implementation. Do not use pre-defined functions. For Python users, you may use libraries minimally to load data or perform basic operations such as matrix multiplications.

Problem 1

(**Sampling**) Before we do any experiment in this homework, make sure that you use gray scale normalized image for your experiment, i.e., your image intensity should range [0; 1]. You may use built-in functions for image normalization. Implement your own function to downsample an image into half size (for x and y directions) by sampling every other pixel. Perform downsampling twice on 'lena.png'. Compare the downsampled images with the original in the same actual size (pixel size for low resolution images should be bigger.) How do they look? Now, perform upsampling of the image by inserting an empty pixel between every and each pixel. Run it twice to get it back to the original size. How are the upsampled images looking? Write your observation in the report.

Problem 2

(**Gaussian smoothing**) Implement a function that takes in an image I, kernel size k and scaling parameter s (i.e., σ) and outputs its smoothed version after Gaussian smoothing, i.e., I_smooth = myGaussianSmoothing(I, k, s). Test it on the given image 'lena.png'. Change the kernel size to k ={3; 5; 7; 11; 51} with fixed s = 1. What changes in the result do you see? Now, change the s = {0.1; 1; 2; 3; 5} with fixed kernel size k = 11. How is the result changing? Explain what you did and a brief report on your observation.

Problem 3

(**Image filtering**) Let's solve the problem that occurred from sampling. Everytime after performing upsampling, perform Gaussian smoothing with k = 11 and σ = 1. Did it get better? How about we perform median filtering? Implement your own median filter and apply it on the same upsampled data. Did it improve the result? Which filtering is better? Discuss your observation in the report.

Problem 4

(**Noise**) On the original 'lena.png' image, let's add some Gaussian noise ~ N(0; 0.1) at each pixel, i.e., I noisy(x; y) = I(x; y) + r where r ~ N(0; 0.1). How is the image looking? Now, let's perform Gaussian smoothing with any parameter you like that works. Did the result improve? Median filter in the previous question probably made you frustrating, but let's give it one more chance. Apply median filtering and see how the result shows. Now, let's go change the type of noise you added. Before you add it to the original image, set the noise to value 1 everywhere where the noise was > 0.2 and 0 otherwise. Apply Gaussian smoothing and see what you get. Apply median filtering as well and see what you get. Discuss your observation in the report.

Problem 5

(**Sobel filters**) Implement a Sobel filter that calculates gradient along x and y directions. To visualize our sobel operator, normalize the gradients so that all the values lie between [0,1]. Your function should look like [mag, ori] = mySobelFilter(I) that takes an image I as an input and output edge response (magnitude) and orientation for each pixel. Try visualizing the result in color by using the magnitude to specify the saturation and value of an image and the orientation to specify the hue. We haven't talked HSV space in the class but can be found at https://en.wikipedia.org/wiki/HSL_and_HSV. Discuss your observation in the report.