# Assignment 1

Due date: 11:59 PM February 19, 2020

## 1 Problems

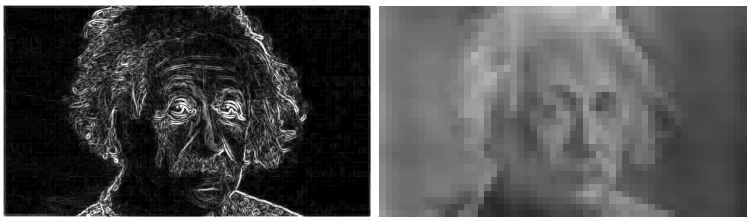
### 1.1 Basic Image Manipulation (3 Points)

First, let's set up some basic image handling functions which will be needed for the tasks that follow.

1. Implement `read_image` and `write_image` according to the function specifications.
2. Implement `display_image`. This function should use matplotlib to display the image.
3. Implement `convert_to_grayscale`. Your method should give the same output as PIL's grayscale conversion, which uses the formula $L = \frac{299}{1000}R + \frac{587}{1000}G + \frac{114}{1000}B$. **DO NOT** use PIL in your implementation.
4. Implement `convert_to_float` according to the function specification.

### 1.2 Convolution (6 Points)

1. Implement `convolution`. This function should take a grayscale image and kernel as input. The image should be zero-padded so that the input and output images are the same size.
2. Implement the kernels `gaussian_blur` and `sobel_filter` in their respective methods. Also implement the difference of gaussians algorithm in `dog`.
3. Implement `visualize_kernels`. This function should read example.png, convert it to grayscale and float-type, and run the functions from part 2 over it. For the gaussian filters, you can use the default parameters. For each function, visualize the result and save it as example_{function_name}.png e.g. example_dog.png.



(a) example_sobel_filter.png (b) example_blurry.png

Figure 1: Example Output Images for Parts 1.2.3 and 1.3.3

### 1.3 Fast Fourier Transform (6 Points)

1. Implement `dft`, which should compute the 2-dimensional discrete fourier transform of the input image. The implementation should be based on the definition given in class and does not have to use the fast fourier transform. Note that the fourier basis from $B_{-M/2,-N/2}$ to $B_{M/2,N/2}$ should be used, as discussed in class. **DO NOT** use the numpy.fft submodule, though you can use it to check the correctness of your implementation.

2. Implement `idft`, which should recover the original image given its fourier transform. The same rules apply as in part 1.

3. Implement `visualize_dft`. This function should read example_small.png, convert it to grayscale and float-type, and run `dft` on it. Try masking out parts of the fourier image and recovering the original image using `idft`. Can you create a blurry version of the original image? Visualize the blurry image and save it as example_blurry.png.

   Note: It may be helpful to visualize the fourier image. For a better visualization, first take the absolute value and log of the fourier image.

## 2 Logistics

- Your implementations go in the `student.py` file. Adding helper functions in `student.py` is ok, but **DO NOT** change any of the existing function signatures.

- **DO NOT** add any new imports to your solution. Using outside libraries outside of the provided ones is not allowed. Moreover, you are not allowed to use functions from the provided libraries that are substantially similar to the functions you are asked to implement (except for image read, writing, and visualization).

- This assignment should be done in teams of 2 people. You are encouraged use the partner finding service on Piazza if you cannot find a partner. For exceptions, please see an instructor.

- **Grading:** This assignment will be graded by an autograder, which will test each function on a set of test cases. The submitted example images will be graded for completion only.
- **Submission:** Please submit your `student.py` file and all example images as a zip file on CMS.