6.s04 Hints for #YesFilter Image Processing

If you are having trouble with this lab, consider the following:

- Remember to ensure the color values produced by each **filter** are in range `[0,255]`, but the output of a 2D convolution may not respect this. In the case of `filter_edge_detect`, clamping the output too early may lead to careless losses: some pixels in the vertical and horizontal components of the image filter may be *negative*.
- Consider writing a set of helper functions to retrieve the height and width of a pixel, and for arbitrary pixels in the image. This is a common operation, and factoring it out will save you some effort.
- Use the web UI to test your convolution.
- Make sure to add new test cases, at the very least the 4 test cases suggested in Part 2 and Part 3.

As a general strategy, the following is a good way to solve this week's lab:

- Write a reusable function `convolve2d(image, kernel3x3)`, where the output is *not* forecd into the [0,255] range. If you feel fancy, consider generalizing it to any `N` x `M` kernel.
- Write a reusable function `combine_images(image1, image2, f)`, where `f` is a **function** of the form `combine_f(pixel1, pixel2)` that describes *how* the two images are to be combined. This is useful for adding, averaging images, to mention just a few.
- Write a reusable function `legalize_range(image)`, that forces each pixel of `image` into integers in `[0,255]`.

Good luck! Sincerely,

your friendly 6.s04 staff