Download and run the homework 3 starter code from the course website before answering these questions. Submit your written answers at the start of class on Thursday.

1. U/V offsets

The `convolve` function in the starter code shows how to implement a convolution in 2D using a quadruply (!) nested `for` loop. In this function, `x` and `y` variables loop over the destination image, and the `u` and `v` variables loop over the kernel weights.

Now consider the `u_offset` and `v_offset` variables defined towards the top of the function.

   a. Try setting one or both of these offsets to zero instead of their current values. How, visually, does the output produced by the `convolve` function change?

   b. Why are these offsets necessary?

2. Border modes

Read the border types documented at the top of `http://docs.opencv.org/3.0-beta/modules/imgproc/doc/filtering.html`, and compare them to the implementation of `handle_border` in the starter code.

   a. Which OpenCV border mode is implemented by this function, as provided?

   b. Modify the function to implement `BORDER_REFLECT_101`, the default border mode used by `cv2.filter2D`. Here are some test cases you might find useful:

      - when `i` is equal to -1, the function should return 1
      - when `i` is 9 and `size` is 10, the function should return 9
      - when `i` is 10 and `size` is 10, the function should return 8
      - when `i` is 11 and `size` is 10, the function should return 7

When you have correctly implemented the function, all 5 comparisons printed out to the console by the program will be `True`; that is, the output of `convolve` will become identical to the output of `filter2D` (if less efficient to compute).

There is no need to turn in your full program source code, just write or type your implementation of `handle_border` in your submitted homework. You shouldn’t need more than 10 lines of code to implement it.

---

1 For the purposes of this assignment, it’s safe to assume that the kernel fits completely within the image, so no need to worry about multiple reflections.
3. Separable filtering

a. Derive an equation to express the number of multiply-add operations performed by our `convolve` function, in terms of the width and height of both the input image and the kernel.

b. In this program, the array `full2` is equal to the array `bxy2`. In light of your answer above, which one would you rather compute, and why?

c. Why are these two arrays equal? Is it because there is something special about the input image? About the Gaussian kernel? Why does the difference between the two approaches for computing this convolution become more important as the size of the kernel increases (say from 5×5 to 55×55)?

Refer to the textbook section 3.2.1 in your explanation.