Math 354, Code Use for the Final Project
Instructor: Dr. Fred Park

1. Given a discrete image u say of size 256×256 on a grid. If you wanted the x-derivative of the function u = u(x,y) and store it as a variable ux, you would use the command: `>> ux = forwardx(u)`.
   This pads the array u by a buffer layer of 2 pixels on each side and then reflects the image onto the buffer layer. It then computes the discrete derivative and then removes the padded layer. In short, it does the following:
   $$u_x(i,j) = \frac{u(i+1,j) - u(i,j)}{dx}$$
   where we assume $dx = 1$ since we are defined on a discrete grid of spacing 1.
   i.e. $u = u(i,j)$, $i = 1:m$, $j = 1:n$, where size(u) = [m,n].
   Note, the command: `>> ux = backwardx(u)` calculates the derivative via backward difference formula:
   $$u_x(i,j) = \frac{u(i,j) - u(i-1,j)}{dx}.$$
   Either one gives a reasonable approximation to the partial derivative in the x-variable. Similar results for the y-derivative via: `>> uy = forwardy(u)` or `>> uy = backwardy(u)` can also be obtained.
   (*Note: I just made the code about 100 times easier to write since you will not need to worry about boundary conditions when calculating numerical derivatives.*)

2. If you want to numerically take the second derivative of u in terms of the x-variable and store it as the variable uxx, you would use the following matlab commands: `>> uxx = forwardx(backwardx(u))`
   You can also use: `>> uxx = backwardx(forwardx(u))`
   If you use either: `>> uxx = backwardx(backardx(u))` or `>> uxx = forwardx(forwardx(u))`
   that would result in numerically instability in the approximation. For example, if I want to calculate the Laplacian of a discrete 2-D function u (which is $\Delta u = u_{xx} + u_{yy}$) and return it as the variable *Lapu*, we would use the following commands:
   ```
   >> Lapu = backwardx(forwardx(u)) + backwardy(forwardy(u))
   ```
   Note that you can also use:
   ```
   >> Lapu = forwardx(backardx(u)) + forwardy(backwardy(u))
   ```
   for reasons explained earlier.

3. Example. If we want to load in the camerman image, calculate the Laplacian of it and then visualize both the image and its Laplacian on the same graph, we would use the following commands:

   ```
   >> f = double(imread('cameraman.tif')); %load in cameraman image
   >> Lapf = forwardx(backwardx(f))+forwardy(backwardy(f)); % calculate Laplacian
   >> subplot(1,2,1); image(f); colormap(gray(256)) % view
   >> subplot(1,2,2); image(Lapf); colormap(gray(256)) %view
   ```

   Behold, in Figure (1) the beautiful images of a cameraman and his Laplacian!

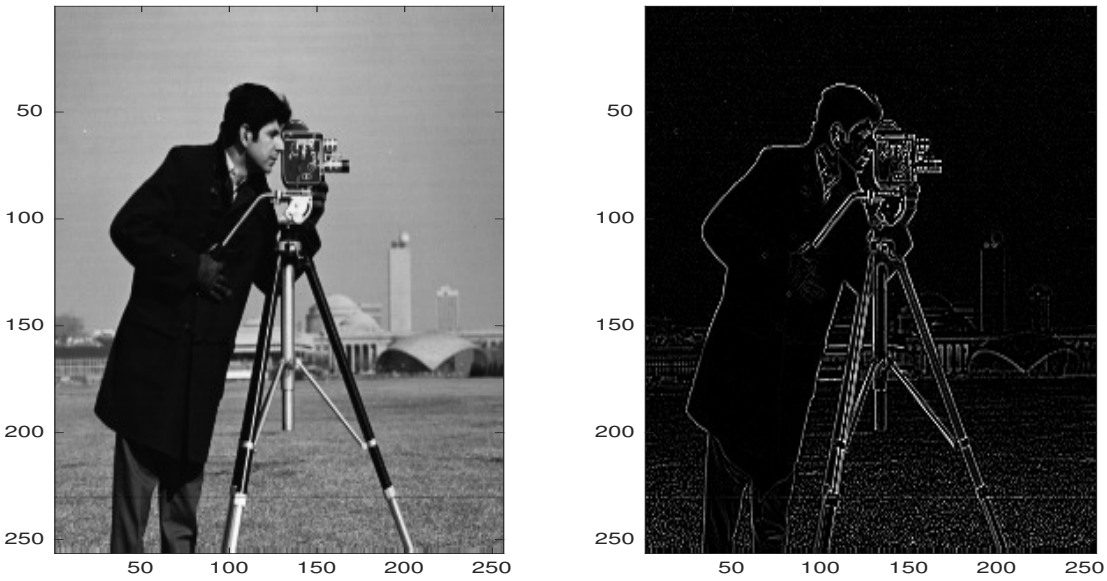

Figure 1: *A cameraman and his Laplacian.*