There are a number of functions that can be used for image processing. In order to be able to think about the code we have to write to perform each of these functions, it is helpful to separate the functions into categories. All the functions in each category share design elements; that is, the basic code is roughly the same in each case. For the examples shown in this document, here is a simple cartoon image:

![Cartoon Image]

1. **Functions that operate on pixels in-place in one canvas.** These functions pick up a pixel, do something to that pixel, and put the pixel back into the canvas at the same place where it picked it up.

   Examples: Negate, RGB, Gray, Monochrome, Brighten, Darken, Increase Red, Decrease Red, Increase Green, Decrease Green, Increase Blue, Decrease Blue, Sepia, etc.

   ![Negate](image1)       ![Mono](image2)

   Requires a new canvas: NO.
2. **Functions that operate on multiple pixels in-place from the same canvas.** These functions move pixels around within the same canvas, often involving swapping the colors of two pixels in different places.

Examples: Mirror, Flip.

![Mirror](image1.png) ![Flip](image2.png)

Requires a new canvas: NO.

3. **Functions that operate on pixels and the pixels in their immediate neighborhood.** These functions look at a pixel and some subset of its eight immediate neighbors (N, S, E, W, NW, NE, SW, SE) to figure out what the new color should be. May also examine more distant neighbors at a commensurate hit in performance (i.e., it slows down). This requires that the new color be placed into a new canvas at the same location of the pixel in the original canvas in order to prevent the new value of the current pixel from influencing the color of as-yet unprocessed neighbor pixels. This also requires that the code make special checks to determine if the desired neighbors are inside or outside of the canvas in order to prevent crashing the program (i.e., detect the edges of the canvas).

Examples: Average, Blur, Emboss, Enhance Focus, Edge Detect, Dithering, etc.

![Blur](image3.png) ![Edge Detect](image4.png)

Requires a new canvas: YES.
4. **Functions that require two canvases of different sizes.** There are two subcategories of this type:

4a. **Destination canvas size is the same as the source, with width and height swapped.** Functions in this category create a new canvas containing the same number of pixels as the original canvas, but the aspect ratio is reversed.

Examples: Rotate Clockwise, Rotate Counter-Clockwise, Transpose.

Clockwise:  

![Clockwise example](image)

CounterClockwise:  

![CounterClockwise example](image)

Transpose:  

![Transpose example](image)

4b. **Destination canvas size is the source canvas size times a scale factor.** Functions in this category also need a new canvas, and while the aspect ratio is the same as in the original canvas the absolute sizes are scaled up or scaled down by some number. Scaling up by a whole number is relatively easy, scaling down or scaling by a non-integer is considerably more difficult.

Examples: Scale Up, Scale Down.

Scale up to 150%:

![Scale up example](image)

Requires a new canvas: YES.