# CS 422/522: Digital Image Processing
# Homework 0 (Fall '12)

1. Download and install *unmscheme* on the computer of your choice. Note: You will also need to install *OpenGL*, *gnuplot*, and *emacs*. Edit your *.emacs* so that the value of *scheme-program-name* is the name (including path) of the *unmscheme* binary on your system.

2. Find a black and white image which you like and figure out how to save it in ASCII *.pgm* format using an image format conversion tool which can read and write this format. Note: *gimp* and *xv* both read and write ASCII *.pgm* images. The Unix *convert* command will also work. Load your image and the Scheme source file "histograms.scm" and display your image's histogram.

3. Write a function *flip-image* which takes an image and reflects it in both the horizontal and vertical dimensions.

4. Find a color image which you like and figure out how to save it in ASCII *.ppm* format. Display your image's color histogram.

5. The ratio of the height to width of most flags is $3/4$. Write a function *flag* which given a *height* returns a color image of a flag of some country. The flags of France and Japan are relatively easy to make using *make-image* and *rgb->color-image*.

6. Write a function *color-image-crop* which acts like *image-crop* but works for color-images.

7. The Mona Lisa owned by the Prado in Spain has long been thought to be a copy of the Mona Lisa owned by the Louvre in France which was painted many years after the original and by an artist with no connection to Leonardo da Vinci. However, recent X-ray studies of the Prado Mona Lisa have revealed that it was repeatedly revised during painting in a way which matches exactly the revision history of da Vinci's masterpiece (also revealed by X-ray studies). The only possible conclusion is that the painter of the Prado Mona Lisa copied da Vinci's Mona Lisa as it was being painted and revised; he was probably a student of da Vinci.

   In addition to the RGB color system with which you are probably familiar, color images can also be represented in the HSI color system (where H

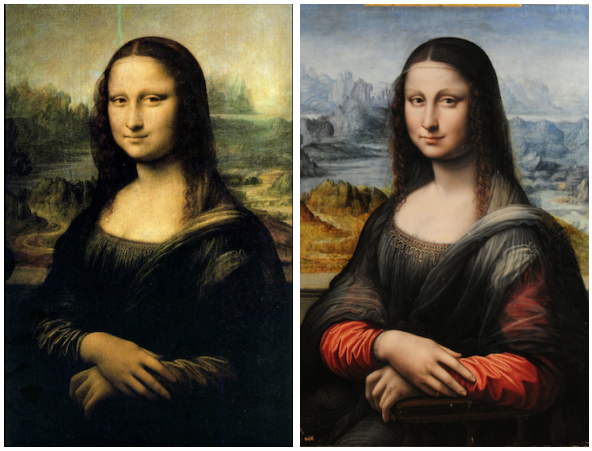

Figure 1: The Louvre Mona Lisa (left) and the Prado Mona Lisa (right).

stands for *hue*, S for *saturation*, and I for *intensity*. We will study color at length later in the semester, for now, it suffices to know that a color image can be decomposed into its HSI component images using the function *color-image−>hsi*, and that HSI component images can be used to construct a color image using the function *hsi−>color-image*.

Although the colors of da Vinci's Mona Lisa have faded with time and yellowed due to the application of many layers of lacquer, the Prado Mona Lisa gives us some idea of what it must have looked like when it was first painted. To accomplish write a function which takes two color images *cim*0 and *cim*1 as arguments and makes a new color image with the intensity of *cim*0 and the hue and saturation of *cim*1. Use this function to make an image which reveals what da Vinci's masterpiece might have looked like when it was first painted.

8. An image can be reduced in size by downsampling in the vertical dimension using *reduce-cols* and downsampling in the horizontal dimension using *reduce-rows*. However, repeated downsampling of an image, *e.g.*, to produce a *thumbnail*, without *smoothing* results in undesirable artifacts due to *aliasing*. We will study this phenomenon in detail later in the course. For now, it suffices to know that an image can be reduced in size in one or both dimensions without aliasing by smoothing prior to downsampling:

```
;; reduce in horizontal dimension without aliasing
(define (reduce-rows x)
  (downsample-rows
   (convolve-rows x #(0.05 0.25 0.4 0.25 0.05))))

;; reduce in vertical dimension without aliasing
(define (reduce-cols x)
  (downsample-cols
   (convolve-cols x #(0.05 0.25 0.4 0.25 0.05))))

;; reduce in both dimensions without aliasing
(define (reduce x)
  (reduce-rows (reduce-cols x)))
```

(a) Write a function *right-split* which takes a square image *im* and an integer *n* and constructs a new image by a recursive process of reduction and concatenation in the horizontal dimension using *reduce-rows* and *left-to-right*. See Figure 2.

(b) Write a function *bottom-split* which takes a square image *im* and an integer *n* and constructs a new image by a recursive process of reduction and concatenation in the vertical dimension using *reduce-cols* and *top-to-bottom*. See Figure 3.

(c) Write a function *corner-split* which takes a square image *im* and an integer *n* and constructs a new image by a recursive process of reduction and concatenation in both horizontal and vertical dimensions. See Figure 3. Hint: I used *right-split* and *bottom-split* in my definition of *corner-split*.

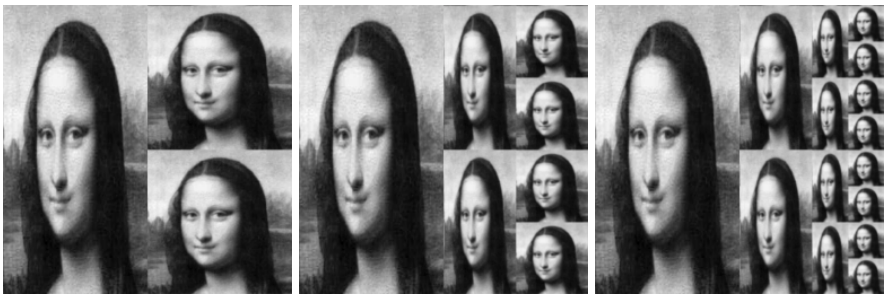

Figure 2: Right splits with $n = 1$, 2, and 3.

Figure 3: Bottom splits with $n = 1$, 2, and 3.

Figure 4: Corner splits with $n = 1$, 2, and 3.