CPS109 Lab 4

Most of the questions in this lab come from Chapter 4 (or earlier chapters) of the course text, Introduction to Computing and Programming in Python, by Guzdial and Ericson. Please put your answers (numbered) in a document and submit it on D2L as a PDF file. Other formats are not accepted.

The **learning objectives** for Chapter 4 are to be able to:

- understand how images are digitized
- identify different models of color, but mainly RGB, the most common for the computer
- to manipulate color values in pictures, increasing and decreasing them
- to convert color to grayscale
- to negate a picture
- to use matrix representation to find pixels in a picture
- to use picture objects and pixel objects
- to use iteration with a for loop to change color values of pixels in a picture
- to nest blocks of code
- to choose between having a function return a value or just have a side effect
- to determine the scope of a variable name

To do:

1) Write a program to set the red, green and blue values to zero. Show your program in your answer document. Try it on a picture ... what is the result?
2) Write a program to set the RGB values to 255. Show your program in your document. Try it on a picture. What is the result?
3) What do each of the following programs do, assuming that the modulo feature is turned off in JES. (To turn it off: Edit > Options > unselect Modulo color values by 256). Your answer for this question can be simple sentences, like:

test1: decreases the red component of the picture by half
test2: ...

```
def test1() :
  f_statue = '/home/eharley/1/statue-tower.jpg'
  picture = makePicture(f_statue)
  for pixel in getPixels(picture) :
    setRed(pixel, getRed(pixel) * 0.5)
  show(picture)

def test2() :
  f_statue = '/home/eharley/1/statue-tower.jpg'
  picture = makePicture(f_statue)
  for pixel in getPixels(picture) :
    setBlue(pixel, getBlue(pixel) * 1.5)
  show(picture)

def test3() :
  f_statue = '/home/eharley/1/statue-tower.jpg'
  picture = makePicture(f_statue)
  for pixel in getPixels(picture) :
    setGreen(pixel, 0)
  show(picture)
```

```
def test4() :
  f_statue = '/home/eharley/1/statue-tower.jpg'
  picture = makePicture(f_statue)
  for pixel in getPixels(picture) :
    red = getRed(pixel) + 10
    green = getGreen(pixel) + 10
    blue = getBlue(pixel) + 10
    color = makeColor(red, green, blue)
    setColor(pixel, color)
  show(picture)

def test5() :
  f_statue = '/home/eharley/1/statue-tower.jpg'
  picture = makePicture(f_statue)
  for pixel in getPixels(picture) :
    r = max(0, getRed(pixel) - 20)
    g = max(0, getGreen(pixel) - 20)
    b = max(0, getBlue(pixel) - 20)
    color = makeColor(r, g, b)
    setColor(pixel, color)
  show(picture)

def test6() :
  f_statue = '/home/eharley/1/statue-tower.jpg'
  picture = makePicture(f_statue)
  for pixel in getPixels(picture) :
    red = getRed(pixel)
    green = getGreen(pixel)
    blue = getBlue(pixel)
    color = makeColor(red, green, blue)
    setColor(pixel, color)
  show(picture)

def test7() :
  f_statue = '/home/eharley/1/statue-tower.jpg'
  picture = makePicture(f_statue)
  for pixel in getPixels(picture) :
    red = getRed(pixel) / 2
    green = getGreen(pixel) / 2
    blue = getBlue(pixel) / 2
    color = makeColor(red, green, blue)
    setColor(pixel, color)
  show(picture)

def test8() :
  f_statue = '/home/eharley/1/statue-tower.jpg'
  picture = makePicture(f_statue)
  for pixel in getPixels(picture) :
    red = getRed(pixel) / 3
    green = getGreen(pixel) / 3
    blue = getBlue(pixel) / 3
    color = makeColor(red, green, blue)
    setColor(pixel, color)
```

```
  show(picture)

def test9() :
  f_statue = '/home/eharley/1/statue-tower.jpg'
  picture = makePicture(f_statue)
  for pixel in getPixels(picture) :
    red = getRed(pixel) * 2
    green = getGreen(pixel) * 2
    blue = getBlue(pixel) * 2
    color = makeColor(red, green, blue)
    setColor(pixel, color)
  show(picture)
```

4) Write a function to blue-ify a face. That is, write a function that accepts a picture as input. If any pixel has a blue value less than 150, then set that pixel's color to white. Show your program. Try it on a picture of a face, in particular, the "jenny-red.jpg" image and report on what you get.

5) That obviously didn't work, but it is interesting, and if you explore the original jenny-red image then you get an idea of what happened. I modified the program to change only the blue for certain values of blue, and below is what I got. Try to do the same or something similar (by trial and error) and show your program.

6) Write a general "blue-ify" function as follows. The function accepts a picture as input. Double the blue value of every pixel and cut the red and green value in half. Include your function and the resulting picture (which you can save using **writePictureTo**(picture, path).
7) Write a single function to change a picture to grayscale and then negate it. For the luminence, use 0.3 * red + 0.6 * green + 0.1 * blue. Include your function and the picture resulting from its application to the caterpillar picture.
8) Rewrite the program from (7) using hierarchical decomposition, where the top-level program calls grayscale(picture) and negate(picture), and you include those two low-level programs following top-level program. Show your programs. Compare the two versions of this program (7) and (8) in terms of readability and efficiency.
9) Write a function to create a lightened grayscale image. First, lighten the image by adding 75 to the red, green and blue components of every pixel. Since high numbers are closer to white, this should make the pixel lighter. Now grayscale the new image. (Make sure modulo is turned off, so the numbers do not wrap when they exceed 255). Use (.3, .6, .1) as the

weights for luminence, as in (7). Show your program and the result of applying it to butterfly2.jpg from the mediaresources image bank.

10) Write a function to create a lightened grayscale image, by using makeLighter(). First lighten the image by using the makeLighter function on each color. Now, grayscale the new image. You can use the grayscale function you wrote for (8). Compare the result to the picture created by the previous problem (9). Show your function and answer: How does makeLighter compare to adding 75 to each of the RGB components of every pixel?