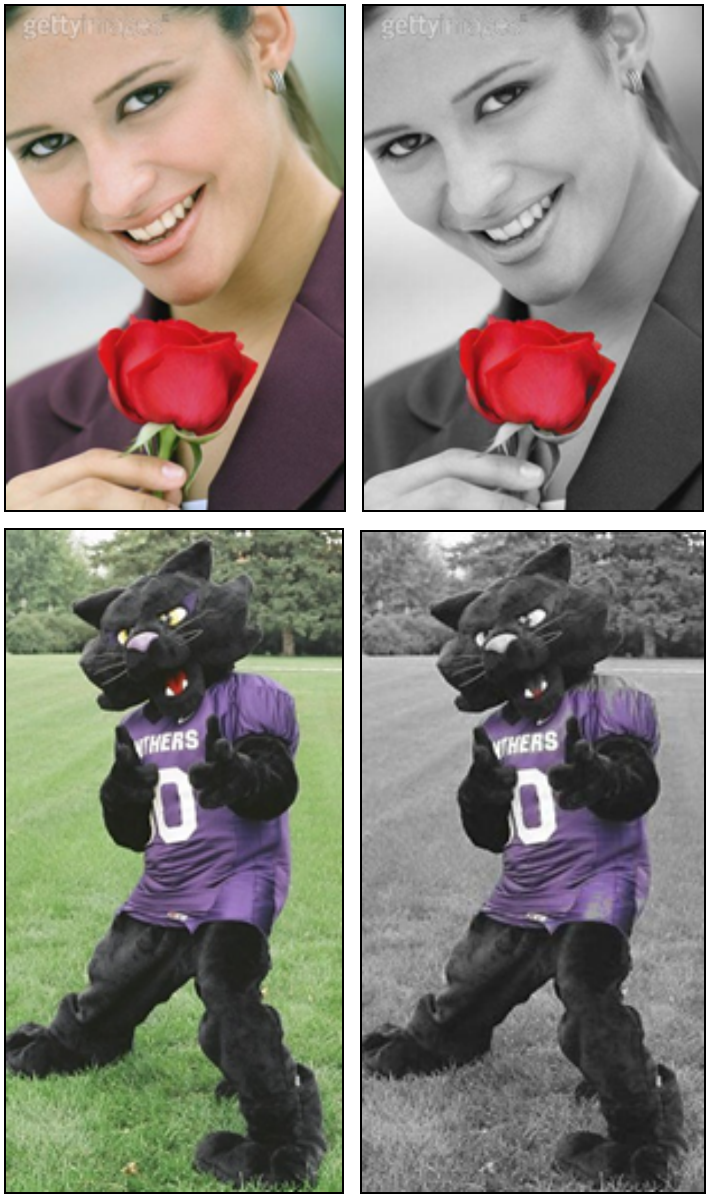
CS 1120: Media Computation Spring 2018
Lab 5: Selective color manipulation

Activity A: Making a color “pop”

Most of you have seen photos that make a color photo nearly completely grayscale EXCEPT for one small feature like a red rose or a pink bow on a dress.

Some examples are below (view these online, a black and white print out won't show the color effect):



In red eye reduction you are looking for every color inside of a box that is very close to a given color and then changing it. Everything in the box but far away from the color, and everything outside of that box, is left alone.

In this color pop problem you are looking for everything inside of a box of a box that is very close to a

given color **and leaving it alone**. Everything in the box but far away from the color, and everything outside of that box, is changed to grayscale. So basically, this technique sounds a lot like the EXACT opposite of red eye reduction. It involves looping through the whole image – but in parts.

Open the rose picture (available on the course website) and, use explore() to answer the following questions:

[Q1] What **x** value defines the ***left edge*** of a box "surrounding" the rose?

[Q2] What **x** value defines the ***right edge*** of a box "surrounding" the rose?

[Q3] What **y** value defines the ***top edge*** of a box "surrounding" the rose?

[Q4] What **y** value defines the ***bottom edge*** of a box "surrounding" the rose?

[Q5] Explore the red values of the rose. Pick a "representative" value from that rose. What are its RGB values?

Make sure you call **setMediaPath()** function from JES (use the command area) – that way you won't have to specify the full path to 'rose.jpg' in your function.

Create a new file for your program: lab5.py. Using the value you just generated, create a function called rosePop() that takes no parameters. The function should immediately open the rose picture (make sure you place it into the same folder as your lab5.py file) and change all of the pixels ***that are not in the box*** around the rose to be grayscale. Your starting code will look like this:

```
def rosePop():
  original =makePicture("rose.jpg")
  roseRed = ???

  # write your loops here that will make everything
  # around the rose be grayscale

  show(original)
  return original
```

There are several techniques that you can use to write the loops that are missing above. I suggest you write a single loop that loops over every pixel in the picture. For each pixel it says "IF the pixel is OUTSIDE of the box, change the pixel to grayscale.

The end result would look like this:

(Note: the black borders are added for emphasis. It is not actually part of the solution.)

Once you have that working you could write a second set of loops that comes after the code you just wrote, but before the show/return code. This code sets up a loop that ONLY looks at the pixels inside of the box around the rose. And this code is truly the opposite of what you did with red-eye reduction. As you loop through those pixels you say “IF the color of this pixel is “far away” (rather than close) to the red of the rose, then change it to grayscale”

However, there is a better way. You can do this in the same loop! Can you see it? look at your IF statement…

Make that change and run your code. If you get everything set up correctly you should be able to get an image similar to the following:

Activity B: Make a collage with copy & paste

Now let's try to make a simple collage by using copy and paste. We will copy the image you have created in the previous activity and paste it four times onto a larger empty "canvas". Write a function makeCollage() that takes no parameters, calls your rosePop() function, and then generates the collage and returns a new picture object. This is quite straightforward; here is what your function should do:

step 1: call rosePop() and assign the return value to a new variable:

```
source = rosePop()
```

step 2: create an empty canvas that is ***twice the width and twice the height*** of the original picture:

```
canvas = makeEmptyPicture(width, height)
```

To calculate width and height, start with getting the source values with **`getWidth(source)`** and **`getHeight(source)`**

step 3: now you need to loop through the source pixels. This time you will need the x and y coordinates of each pixel. Recall that you can use this function: **`getPixel(picture, x, y)`**

You will need a nested for loop:

```
for x in range(width):
     for y in range(height):
          pixel = getPixel(source, x, y)
```

Inside the loop, get the color of the current pixel: color = getColor(pixel)

Then, set the corresponding pixel of the target canvas to that color.

However, since we are creating a collage, you will need to ***set the color to four different pixels*** on the target canvas to achieve the following result:

Can you figure out what pixels you need to set? Think about the width and the height of the source – can we use those values to calculate the x and y values of the four target pixels?

After the loop, return the new canvas: **`return canvas`**

When you call your new function, assign its return value to a variable, and then write it to a new file:

```
collage = makeCollage()

writePictureTo(collage, "collage.jpg")
```

And you're done! Save your lab5.py file and submit it to eLearning. Make sure to save your solution for future reference.