CptS 111 Lab #10

In this lab, you will perform basic graphic manipulations on portable pixel map (PPM) image files. In order to complete this lab, your TA will need to cover:

1. String splitting using .split()
2. String joins using .join()
3. Modulo arithmetic

Prerequisites

Before beginning this lab, you will need some sort of imaging software that can open PPM-based images. I would recommend Ifranview for Windows as it's free and lightweight. I believe that OSX supports PPM natively, but if I'm wrong, you can download ToyViewer for free.

Before We Begin: DON'T PANIC

In this section, I describe the PPM format in detail. **DON'T PANIC!** You don't need to perfectly understand the PPM format in order to complete this lab. Remember, you can always ask your TA if you get stuck.

PPM Image Format

The PPM image format is encoded in human-readable ASCII text. It might be helpful to read over the formal PPM specification document. A PPM document has two pieces: the header and the body. The header is always the top three uncommented lines in the file:

```
P3
4 4
255
```

The first line specifies the type of image that is contained within the file. We will always use the "P3" specification. The second line specifies the number of columns and rows present in the image. In this example, we have a 4 pixel by 4 pixel image. The final number indicates the maximum value for each red, green, and blue (RGB) element in the picture. Having a max value of 255 is quite common and is the value that we will always use. Below the header is the body of the PPM file. Each pixel has a red, green, and blue value. For example, the content of our 4x4 image might be:

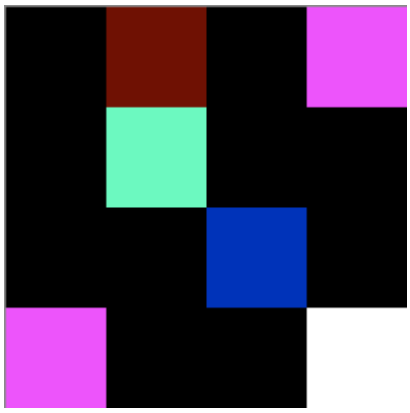
```
0  0  0       100 0  0       0  0  0       255   0 255

0  0  0       0 255 175      0  0  0        0    0  0

0  0  0       0  0  0        0 15 175       0    0  0

255 0 255     0  0  0        0  0  0       255  255 255
```

With these values, the pixel in the first column and row has a RGB value of (0,0,0), which equates to black. The last column in the first row has an RGB value of (255,0,255), which equates to white. A blown-up image containing these values would look like:



Task

Your task for this lab is to write an image effect that removes all green from a given image. Note that this can be accomplished by simply setting the green value in each RGP triplet to zero.

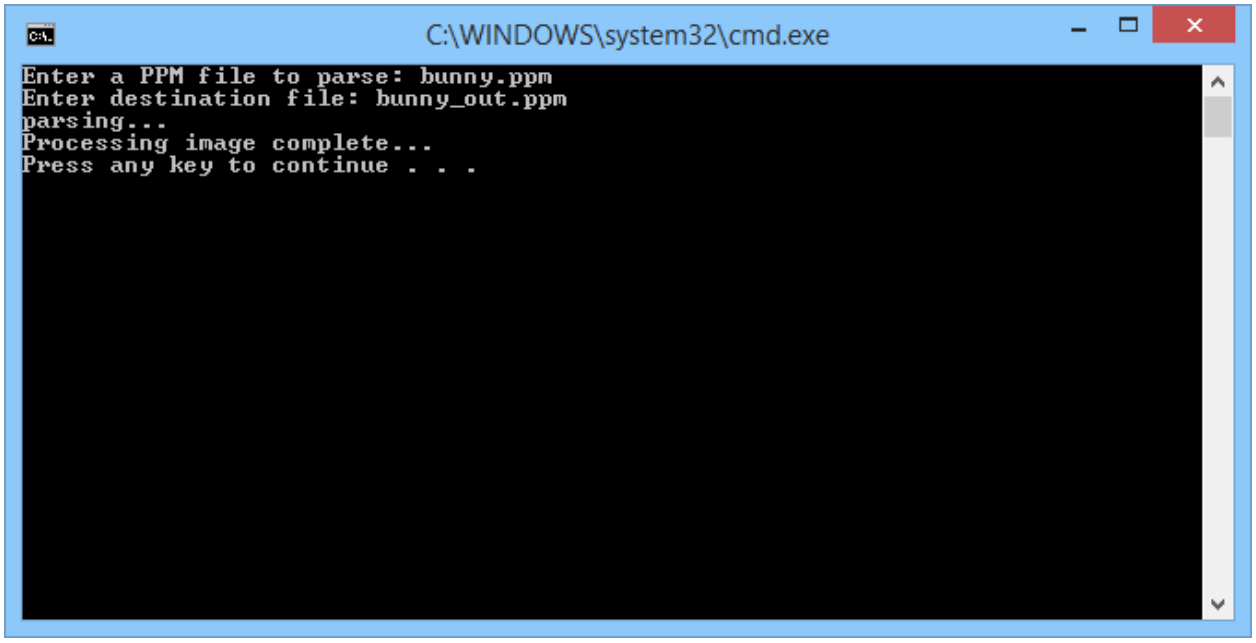
Algorithmic Steps

To complete this lab, it might be helpful to follow my algorithm:

1. Prompt user for input and output files, open accordingly.
2. For the first three lines in the input file:
    a. Use .readline() to get the line
    b. Write that line to the output file (remember to use end='' in your print!)
3. For the remaining lines in the file
    a. Split each line into an individual number using .split()
    b. Set every green value to 0. Remember that every three values represents a complete RGB series. As such, if you use a counter variable to alter ever 2nd value, you will only touch the green values.
    c. Write this modified sequence to your output file

Sample Output

Below is a screenshot of my lab along with the input and output files:



Note that the "parsing" stage may take a while (several seconds).

Bunny.ppm

This is the original bunny file:

Bunny_out.ppm

And the image with green removed: