Short Program 2: .ppm image converter

Due: Friday, Sep 9 at 11:59 pm

Overview

In this assignment you will extend the .ppm header reader so that it can read a color .ppm image and convert it to grayscale. You *must name* your source code *sp2.c*. You may use the solution to short program 1 as a starting point. The source code is in

/home/westall/class/215/assns/sp1data/sp1.c

Immediately following the the new line character that follows the 255 in the .ppm header is the image data. For color images of type P6 it is encoded as follows:

```
RGBRGBRGBRGBRGBRGBRGBRGBRGBRGBRGBRGBRGBRGBRGBRGBRGBRGB...
```

RGB represents the *red*, *green,* and *blue* intensities of the three bytes making up each pixel. A value of 0 represents no intensity (black) and 255 is the most intense representation of the color. The bytes making up the image are do not contain any embedded delimiters. Thus the remaining size of the file is *image-width* * *image-height* * *3* bytes. The pixels defining the image are stored left to right, top to bottom.

The output format that we will use is the P5 type of .ppm file. This is sometimes also called PGM (Portable Grayscale Map). Here is a typical format:

```
P5
800 600
# 800 is the width. 600 is the height.
255
YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY
YYYY . . .
```

In the gray scale image the value *Y* represents its *luminance* which is a measure of how bright it is. As with color images, 0 represents black and 255 the most intense white available on the display.

Since a pixel is represented by a single byte in this encoding, there are *image-width*image-height* bytes following the header of a PGM file.

Program operation

After processing the .ppm header of the input file,

1. use two calls to the *malloc()* function to allocate space for the input image and the output image. Note that the size of *the input image is 3 times larger than the output image.*

2. Use a single call to the *fread()* function to read the *entire* input image from the *standard input.* Any attempt to prompt for a file name and/or explicitly use *fopen()* to open the input file will receive a *20 point deduction*!
3. For each three bytes comprising a pixel of the input image create a single output image pixel using the formula: Y = .3R + .59G + .11B where (R, G, and B) represent the red, green and blue intensities of the input pixel.
4. Use a single call *fprintf(stdout, ...)* function to create the output (P5) .ppm header.
5. Use a single call to the *fwrite()* function to write the grayscale image to the standard output.

Additional notes

Use the *xv* program to ensure that your output image looks correct.

Deductions will be made if warnings remained when compiled with -Wall

How to submit your program:

NOTE: This procedure has NOTHING in common with "handin" nor "sendlab"
Do NOT even TRY to think about how they fit into this procedure because
THEY DON'T!!

<<<Do NOT turn in any image files, core files, makefiles etc.>>>

You must turn in 1 file: sp2.c. Use the same procdure that was described in the sp1 assignment to turn the program in.

After you think you have turned your programs in, its a good idea to

```
cd /local/jmw2/215/sp2/wjsmith
```

and make sure your files are their and they still compile and work correctly.