Programming Assignment 2, Fall 2017

OBJECTIVE

The objective of this assignment is to give you more practice with Unix, with using 2D arrays, and with binary representation of data.

ASSIGNMENT SUBMISSION

To get credit for this assignment, you must

- ✓ complete Unix tutorial (25%)
- ✓ write a program in C (75%)
- ✓ submit your files through Canvas exactly as instructed (naming, compatibility, etc.)
- ✓ submit your assignment on time

UNIX TUTORIAL

Go to Code Academy website (https://www.codecademy.com/courses/learn-the-command-line) and complete the other two modules from *Learn the Command Line* tutorial: *Redirecting Input and Output* and *Configuring the Environment*. After you complete both tutorials, take a snapshot that shows your name and either the completed course or the two tutorials - upload to Canvas as `jpg, gif, png` or `pdf`. Your file is to be named `<yournetid>_badges2.<extension>`. If you can find some other way of showing completed lessons that includes your name, you may take a snapshot of some other screen.

C PROGRAM

Problem Statement

Create a program that reads 24-bit bmp files and generates new versions of the original image. The new versions to be supported are:

- a brightened copy of the original
- a copy of the original with increased contrast
- a copy of the original rotated by 180 degrees (note – this is not the same as flipping)
- scaled down repeated copy of the original

You need to follow the algorithms explained below while processing the images.

Your program is to prompt for the name of the input file and its pixel dimensions. Your program has to run exactly as shown below and follow the same input format. Note that although a file will have a bmp extension, a user will only enter the part of the name that precedes it; then the user is to be prompted for height and width (in this order). This is a sample run of the program (bold and italics indicate user entries).:

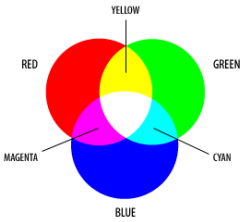
```
---
Enter the filename: test1
Enter height and width (in pixels): 160 240
Done. Check the generated images.
---
```

Note that the generated files should be named:

```
copy1.bmp
copy2.bmp
copy3.bmp
copy4.bmp
```

BMP files

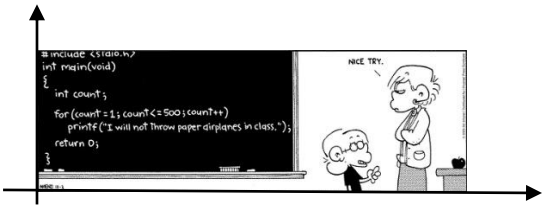
A 24-bit bmp file is capable of storing 2d images. A file consists of a 54 byte header and the rest of the file is pixel information. Each pixel is represented by 3 bytes, where each byte contains the information for blue, green, and red hues – in this order. Each hue can be of value 0..255, where 0 signifies the complete lack of that color, while 255 signifies its full saturation. New colors are generated by mixing different intensities of this basic color palette. All together 256^3 = 16,777,216 colors can be represented using this scheme.



Note that the image height and width in pixels correspond to how we store information in a 2D array. In the bmp format the pixel (0,0) is located at the bottom left (origin of the plane) at locations (0,0), (0,1), (0,2).

Width (x 3)

height	0	1	2	3	4	5	.....				N
height-1	Pixel 1 B	Pixel 1 G	Pixel 1 R	Pixel 2 B	Pixel 2 G	Pixel 2 R	.....		Pixel N B	Pixel N G	Pixel N R
...											
2											
1											
0											



A sample file `bmpchange.c` is provided to show how to read and write such files, and how blue and green components of each pixel could be switched. You are to use this file as the basis of your program.
A few small bmp test files are provided with this homework as well.

Pixel Manipulation Algorithms

Brightening Algorithm
For each pixel, adjust each of its BGR values up by 50. Make sure you do not go above 255 though.

Contrast Algorithm
For each pixel calculate its new BGR value using the formula:

new_blue_value = contrast_ratio * (old_blue_value - 128) + 128
new_green_value = contrast_ratio * (old_green_value - 128) + 128
new_red_value = contrast_ratio * (old_red_value - 128) + 128

where contrast_ratio = 2.9695
If the resulting new B/G/R value > 255, make the color 255. If the resulting new B/G/R value < 0, make the color 0.

180 Rotation
180 degree rotation means that each column of pixels is reversed and each row of pixels is reversed. Note that in the matrices shown below, letters denote whole pixels and not their BGR components.

Before

a	b	c
d	e	f
g	h	i
j	k	l

After

l	k	j
i	h	g
f	e	d
c	b	a

Before

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

After

p	o	n	m
l	k	j	i
h	g	f	e
d	c	b	a

Scale down repeat

To create one scaled down version of the image:

For each pixel in the image, calculate its new location by diving its coordinates by 2. As a result, some pixels in the original will be lost.

Then, take the new scaled down version and copy it to the 3 remaining portions of the image.

Note that in the matrices shown below, letters denote whole pixels and not their BGR components.

Before

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

After

f	h	f	h
n	p	n	p
f	h	f	h
n	p	n	p

Before

a	b	c
d	e	f
g	h	i

After

h	i	h
e	f	e
h	i	h

Other Specs

- Your program is to be contained in a single c file named ***pr2.c***
- Your program has to follow basic stylistic features, such as proper meaningful variable names, etc.
- Overall, your program is to use at most two 2D variable length arrays: one that contains the original image and the other that contains the copy of the image
- If you want to copy an entire matrix at once, use *memcpy* function to do so instead of copying cell by cell
- **Your program must compile in gcc gnu 90 – programs that do not compile will receive a grade of 0**
- Your program has to follow basic stylistic features, such as proper indentation, whitespaces, meaningful variable names, etc.
- Your program should include the following comments:
    - Your name at the top
    - Whether you tested your code on the cssgate server or Ubuntu 16.04 LTS Desktop 32-bit
    - Comments explaining your logic
    - If your program does not run exactly as shown above, explain at the top how to run your program – you will not receive full credit but at least you may receive some credit rather than none

Extra Credit (15%)

Provide some other interesting and relatively complex manipulation of the original image – explain in comments at the top of your code. Simple color manipulation does not qualify for extra credit. In order to earn extra credit, image manipulation needs to either involve a relatively complex algorithm that manipulates colors or provide size related scaling that requires adjustments to the image header.

Program Submission

On or before the due date, use the link posted in Canvas next to *Programming Assignment 2* to submit your tutorial snapshot and your C code. Make sure you know how to do that before the due date. Valid documentation file formats are: pdf, jpg, gif, png. Valid program format: a single file named *pr2.c*