CpSc 1010 Spring 2018
Programming Assignment #3
ppm Image Manipulation

Due 11:59 pm Sunday, April 22nd

1. Overview

The purpose of this assignment is to give you practice with multi-module programs and makefiles, header files, using file pointers, and command-line arguments as well as redirection, along with the topics from the first two assignments.

2. Notes on Collaboration

You are required to work individually on this assignment. Please do NOT consult *anyone* other than your instructor or the lab assistants on *any* aspect of this assignment. Other tutors that are available may be able to help with more general C concepts that you may be struggling with throughout the semester, but NOT with questions specific to the assignment.

Review the section on "Academic Integrity" on the Course Syllabus for the policy on what happens when copying or sharing code.

For any of the programming assignments in lecture and/or lab, it is considered cheating to do any of the following:

- discuss in detail the C code in your program with another student
- show or share the C code in your program with another student
- use C code obtained from another student, or any other unauthorized source, either modified or unmodified (each student is responsible for protecting his or her files from access by others)
- use re-engineering tools
- submit work of others, from the Internet or any other source, even if attributing the work to others

Any programs that closely resemble each other after running through a plagiarism detection tool will be sent immediately over the office that deals with academic dishonesty, no questions asked.

3. More Assignment Specifics & Example Output

GENERAL DETAILS

For this assignment, you will write a program that manipulates a ppm image in the following ways: mirror image (rotated along a vertical axis); upside down image (rotated along a horizontal axis); and a gray scale image with a slightly orange tint.

Your program will consist of multiple source files, a header file, and a makefile. You should have the following files with the specified functions (along with any other additional helper functions that you may want to also create):

`mainDriver.c` – contains the `main()` function, where:

1. the file entered at the command line will be opened
2. `parseHeader()` will be called
3. the 2-D array will be declared
4. `getImage()` will be called
5. `printMenu()` will be called
6. depending on the value returned by `printMenu()`, one of the following functions will be called:
    a. `mirror()`
    b. `flipHoriz()`
    c. `grayscale()`
7. `printHeader()` will be called
8. `printImage()` will be called

`menu.c` – contains the `printMenu()` function, which will present the menu to the user and return the chosen value

`parse.c` – contains the `parseHeader()` and `getImage()` functions

`print.c` – contains the `printHeader()` and `printImage()` functions

`mirror.c` – contains the `mirror()` function

`flipHoriz.c` – contains the `flipHoriz()` function

`gray.c` – contains the `grayScale()` function

`transform.h`

`makefile`

Notice from above, only one function returns something – the `printMenu()` function returns an `int`; the other functions do not return anything. Also, `printMenu()` doesn't require any arguments.

The only functions that need the file pointer as one of the arguments are the `parseHeader()` and `getImage()` functions.

The other functions need at the very least, the rows and columns, and most also need the 2-D array as well.

The 2-D array should be an array of pixels, defined by a structure. You can typedef the structure or not, that's your choice, but it should be called `pixel` and needs to contain 3 `unsigned char` data members, one for each pixel value (red, green, and blue).

The prototypes for all your functions should be in your header file, along with any other #include statements and your structure definition.

Your program will use command-line arguments (to specify the input image file) as well as redirection (for the newly created image). When run, a menu will be presented to the user to choose which manipulation to perform. An example of what it should look like when run is below:

```
[15:45:27] chochri@koala4: [5] transform tiger.ppm > img.ppm

1. original image
2. mirror image
3. upside down image
4. gray scale image

MENU CHOICE:   2
```

At that point, in the above sample run, the specified output image called `img.ppm` should have been created, which you can then view with an image viewer/editor, such as gimp.

If your program does not work as specified, you will lose points accordingly.

PHASE I

The first thing you will need to do is to be able to parse out the dimensions of the image from the header portion of the image file. This is what you will do in lab 11, which you will then incorporate into your PA3 program. Because ppm image headers can vary, this can be the trickiest or most frustrating part of the assignment. Once you are able to parse the header information, you should test it by adding the `getImage()`, `printHeader()` and `printImage()` functions. When that works, then you can move on to phase II.

PHASE II

For this phase, you can add the `printMenu()` f unction, testing out that it works by printing a message to the screen. Once that works, you can delete that print statement and add the functions for the image manipulations, one at a time, checking that each one works before moving on to the next one. Each of these functions requires slightly different logic. It may be helpful to draw out on paper a smaller 2-D array example, perhaps something like a 10 X 10 matrix.

4. What to Hand In

Your files MUST BE NAMED as specified above on page 1 and all submitted to the handin.cs.clemson.edu page by midnight Sunday, April 22nd.

The source code will be evaluated not only on its correctness, but also on its adherence to the coding style standards "Programming Assignment Requirements" provided on Canvas. You should also take a look at "Formatting Code Examples" also provided on Canvas. Don't forget to use meaningful variable names, and don't forget about indentation, comments, and lines not exceeding 80 characters (including spaces).

If you use vim as your editor, remember that you can create a .vimrc file as explained on one of the last slides from the chapter 2 slides. That will automate and make consistent the indentation for you.

5. Grading Rubric

If your program does not compile on our Unix machines or your assignment was not submitted on time, then you may receive a grade of zero for this assignment. Otherwise, points for this programming assignment will be earned based on the following criteria:

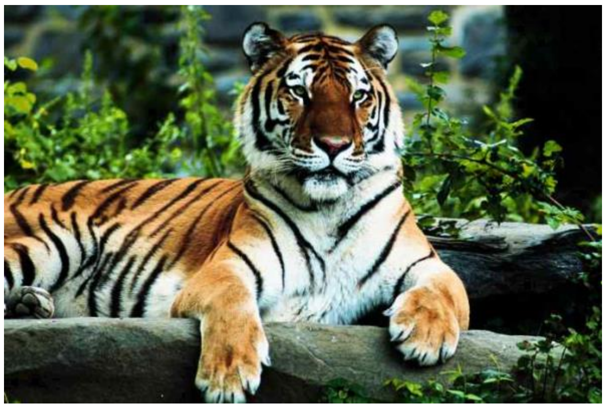
60	functionality and adherence to specs in the write up or explained in class for the 7 source files
5	header file
5	makefile
5	uses file pointer to open image
5	uses redirection for the manipulated image
5	uses a structure for the pixel
5	uses a 2-D array of pixels
10	proper code formatting

OTHER POSSIBLE POINT DEDUCTIONS (-5 EACH): There could be other possible point deductions for things not listed in the rubric

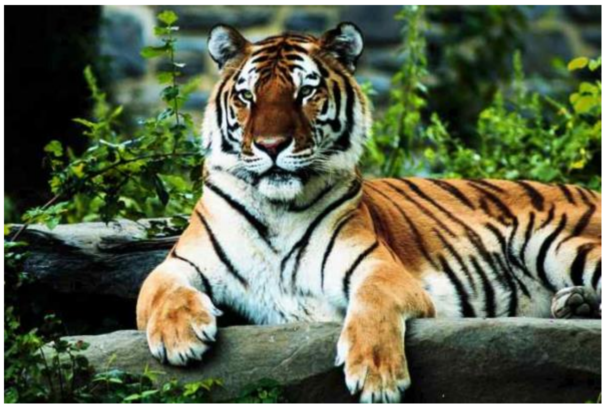
- warnings when compiled
- use of break not in switch statements
- naming the files incorrectly
- missing comments before each function as shown in example code above
- missing return statement at bottom of main() function
- missing header comments at the top of EACH source file
- etc.

6. Sample Images

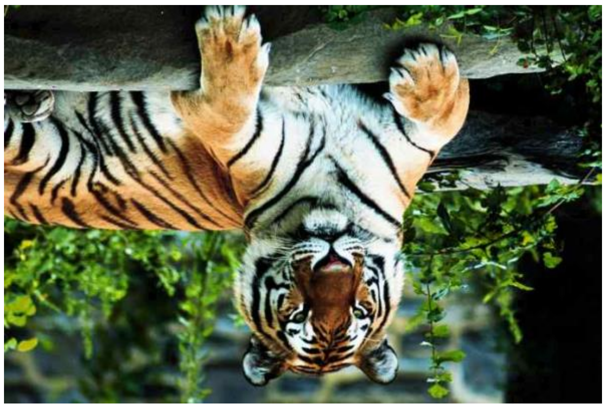
GIVEN IMAGE



MIRROR IMAGE



UPSIDE DOWN IMAGE



GRAY SCALE WITH SLIGHTLY ORANGE TINT