Lab 7: Intro to PPM

CPSC 2310 – Spring 2024

Due Date: Tuesday, March 12, 2024 @ 11:59 PM

I am setting the due date to Tuesday since we will have an exam on Monday. There will not be an extension on this lab, so DO NOT wait until the last minute to start working on this lab.

This lab is an individual assignment. You are not allowed to work with anyone on this project. If you need help contact one of the lab TA's or Dr. Feaster.

Overview:

You are going to write a program that will read the header of a ppm image, ignoring all header comments. Your program will also read the pixel values of the image. Lastly, your program should write the image out to another ppm file (minus the comments in the header).

I will provide a file, named ppmUtil.h, that will contain two structs, one to represent the pixels of a ppm image and one to represent the data that makes up the header of a ppm image. The ppmUtil.h file also contains prototypes of functions you are going to implement.

You will provide ppmDriver.c and ppmUtil.c files.

I will provide 2 images for testing. Your program should be able to recognize and ignore the comments from the ppm images provided. Your program should read the header information and the pixel values of the image, storing each. Then use the stored information to create a new ppm image that has no comments in the header.

You are required to use **fread and fwrite** when reading and writing the **pixel** data. You will use **fscanf and fprintf** when reading and writing the **header** information.

Below briefly describes the ppmDriver.c. You will provide the ppmDriver.c file.

The driver should have minimal amount of code in it.
It should create and open the needed file pointers. There will be two file pointers - one for the input ppm file and one for the output ppm file. The name of the files will be provided as command line arguments. The first will be the input file with the second being the output file.
Your program should create a variable of type header_t and a pointer of type pixel_t.
The driver should then call the function **read** then **write**.
The driver should then call the function to give the memory back to the OS and close the file pointers.

The following will **briefly** describe the function parameters and what each function does. You are required to implement each of these function. The function prototypes will be included in the ppmUtil.h file. You are not allowed to change the function prototypes.

pixel_t* read(FILE*, header_t*);
FILE* - Represents the input image.
header_t – represents the header data of the input image.

Calls readHeader.
Calls readPixels.
Returns the pixel_t pointer created for readPixels.

void readHeader(FILE*, header_t*);
FILE* - points to the input file.
header_t* - is used to store the information read from the input image.
This function will read in each of the 4 parts of the header of a ppm image. You should use fscanf when reading. Read one part of the header then call the function to check for comments. (We discussed in class the rules for comments in a ppm header.). If you missed the class you should come to my office or ask your TA for help.

pixel_t* readPixels(FILE*, header_t);
FILE* - points to the input file.
header_t – contains the information needed to read in the pixels for the image.

You will need to call the function allocatePixMemory to allocate the memory for the pixels.
You will then use **fread** to read the pixels storing them in the allocated memory, then return the pixels.

void write(FILE*, header_t, pixel_t*);
FILE* - represents the output file.
header_t - represents the header data of the output image.
pixel_t – contains the pixel data to be written.

Calls writeHeader;
Calls writePixels;

void writeHeader(FILE*, header_t);
FILE* - points to the output file.
Header_t – contains the information needed to write the header for the output image.

Use fprintf to write the header information to the output file.

void writePixels(FILE*, pixel_t*, header_t);
FILE* - points to the output file.
pixel_t* - contains the pixels read from the input image.
header_t – contains the header information from the input image.

Use **fwrite** to write the data from the input image to the output image.

pixel_t* allocatePixMemory(header_t);
header_t – contains information about the header.

Allocates the memory for the pixels of the image and returns the allocated memory.

void freeMemory(pixel_t*);
pixel_t* the memory that contains the pixel information.

Give the memory back to the OS.

void ckComment(FILE*);
FILE* - points to the input file.
Checks for and ignores the comments in the ppm header.

Other Requirements:

1. Add header guards to ppmUtil.h
2. Add comments to ppmUtil.h – Example given below
3. Create a makefile
4. No lines of code/comments should be longer than 80 characers
5. Code should be neat and easily readable.

Here are some guidelines for documenting the code in your assignment.
Before each function in the .h file, you should have a detailed description of what the overall function does. To borrow from another student's code, here is an example of overall function description. Your description should be more than just a word or two.

```
/* Parameters: img - image_t pointer array holding the image data for
 *                   each of the input files
 * Return:       output - image_t struct containing output image data
 * This function averages every pixels rbg values from each of the
 * input images and puts those averages into a single output image …
 */
```

Also, if you include comments in the body of the function (and you should) they should be placed above the line of code not beside the code.
Example:

Good
```
//This is a comment
if(something)
{
     do something;
}
```

Bad
```
if(something) //This is a comment
{
     do something;
}
```

Submission Information:
You will submit the following files in a **zip folder (NOT tarred) called lastname.zip** (where lastname is your last name) **to canvas**:
ppmDriver.c
ppmUtil.c
ppmUtil.h
makefile