The purpose of this assignment is to give you additional practice in working with functions, dynamic arrays, and images.

For this assignment, you may work with a partner. If you choose to work with a partner, one of you should send me both names by Friday, February 23, 11:59 pm.

**Problem statement:** An interesting and fun effect that we can apply to images is mirroring. For this assignment, you are to mirror a color image along a vertical axis. Imagine that you have a mirror, and you place it on a picture so that the left side of the picture shows up in the mirror. That is the effect that we will implement in this assignment.

To do this, we want to copy the input pixel in the first column and first row (0,0) to the output pixel in the first column and first row **and** to the output pixel in the last column and first row (width - 1, 0). We will copy the input pixel in the the second column and first row (1, 0) to the output pixel in second column and first row **and** to the output pixel in the second to last column and first row (width - 2, 0). We will continue to do this until we reach the middle of the picture (width / 2), including width / 2. Remember that the index for the last column is one less than the width; therefore each time through the loop, we are copying from the (x, y) input pixel to the (x,y) output pixel **and** the (width - x - 1, y) output pixel.

Below are examples of images mirrored along a vertical axis.

|  |  |
| --- | --- |
| https://people.cs.clemson.edu/~rlowe/cs1070/programs/spr18/prog2/flower.jpg | https://people.cs.clemson.edu/~rlowe/cs1070/programs/spr18/prog2/mirror-flower.jpg |
| https://people.cs.clemson.edu/~rlowe/cs1070/programs/spr18/prog2/beach.jpg | https://people.cs.clemson.edu/~rlowe/cs1070/programs/spr18/prog2/mirror-beach.jpg |

While mirroring is probably mostly used for interesting effects, occasionally, it has a purpose. Consider a picture of the Temple of Hephaistos, which is in the ancient agora in Athens, Greece. The pediment of the Temple of Hephaistos was damaged, as indicated by the following image on the left. Vertically mirroring the image along the mirror point for the damaged pixels produces the image on the right.

|  |  |
| --- | --- |
| https://people.cs.clemson.edu/~rlowe/cs1070/programs/spr18/prog2/temple.jpg | https://people.cs.clemson.edu/~rlowe/cs1070/programs/spr18/prog2/mirror-temple.jpg |

**Program input**

For this assignment, the input file name and the output file name will be given as command-line arguments.

**Getting the files for this assignment**

1. Use *mkdir* to create a prog2 directory for program 2. You may name your directory whatever you want.
2. **cd** to the prog2 directory.
3. Download the [prog2.tar](https://people.cs.clemson.edu/~rlowe/cs1070/programs/spr18/prog2/Public/prog2.tar) file to your prog2 directory. You can save the tar file from your browser, or you can directly copy this file over the network using the scp utility:

***scp access.cs.clemson.edu:/home/rlowe/public\_html/cs1070/programs/spr18/prog2/Public/prog2.tar .***

    Note the "**.**" at the end of the scp -- it is important -- meaning extract to the "current directory".

1. Use the command below to extract the files in prog2.tar

***tar -xvf prog2.tar***

After you untar the files, you may delete prog2.tar

Your prog2 directory should contain the files

*image.h*, *getnum.c, getimage.c, newimage.c outimage.c, mirror.c, prog2.c*, and *makefile*. It should also contain two .ppm files for testing your code.

***image.h***> contains struct definitions and function prototypes for this assignment.

***getnum.c*** contains the *getint()* function that is needed by the *getImage()* function.

***getimage.c*** contains the function, *getImage()*, which opens the input file, reads the image data, creates the *inImage* struct, closes the file, and returns a pointer to *inImage*.

***newimage.c*** contains the stubbed function, *newImage()*, to be completed for this assignment. The function must malloc memory for a new image\_t struct, and assign appropriate values to the data members. **Note:** the function must also malloc memory for the image itself and assign the pointer to the pixel\_t \*image pointer in the image\_t struct.

***outimage.c*** contains the stubbed function, *writeImage()*, to be completed for this assignment. writeImage() must do the following:

1. open a file using the name passed in as a parameter.
2. write the header, i.e., P6 width height brightness, to the file
3. use *fwrite()* to write the image data to the file.
4. close the file

***mirror.c*** contains the stubbed function, *mirror()*, to be completed for this assignment, as described above.

***prog2.c*** is the test driver for program 2.

The *makefile* helps you to build and manage the assignment.

**Compiling and Testing Your Program**

To build the executable file, you may use

make   
OR   
gcc  -std=c99  -Wall  -o mirror  \*.c

The command to test your program is

./mirror  inputfilename  outputfilename   
e.g.   
./mirror  flower.ppm  mirror1.ppm

**Submitting your files**

1. If you worked with a partner, **only one** of you should submit the assignment. All files should contain the name and username for both of you.
2. First, if you are not in your program 2 directory, move to it and *make clean*
3. tar all files to  myprog2.tar:    *tar  -cvf  myprog2.tar  \**   
   There is a 10-pt penalty for using the wrong file name.
4. Submit ONLY myprog2.tar via [handin](https://handin.cs.clemson.edu/)

It is your responsibility to make sure you have submitted the correct file. Check your Handin folder after submitting the file. There is a 10-point penalty for using the wrong file name and a 10-point penalty for submitting the file to the wrong folder.

**Requirements**

1. For this program, you must work only with your partner. You may discuss the problem with other classmates, but at no time should you discuss code in any form. Any evidence of cheating will result in a grade of **-100** for all students involved. If you have questions, you should check with your partner, me or a lab instructor.
2. Your program must adhere to the [program standards](https://people.cs.clemson.edu/~rlowe/cs1070/programs/standards.html) and [program requirements](https://people.cs.clemson.edu/~rlowe/cs1070/programs/requirements.shtml). Violations will lead to deductions. In particular,
   * no more than one statement may be written on a single line; this includes declarations.
   * use reasonably descriptive names for variables and functions. no abbreviations allowed, unless universally known.
   * code lines should not extend beyond column 80.
   * indenting should be consistent with logical nesting; indent 3 - 4 spaces.
   * do not use a **break** or a **return** statement to prematurely exit a control structure.
   * a brief description must precede each function.
   * no function should be longer than 30 lines of code, excluding whitespace.
   * your program should compile without any warnings with gcc -Wall
   * diagnostic / debug prints should be disabled/deleted in the final submission.
3. The program is due by 11:59:59 pm. Programs must be submitted on time. The late pass policy applies
4. Programs must compile. Programs that do not compile will receive a score of **zero**.