Please read this entire prompt!

Assignment:

1) Change your Image struct to a class.
2) Write additional functions to manipulate images.
3) Confirm that your program creates the right output.

== 1. Change your Image struct to a class ==

Put your Image in a class.
The class should have three constructors (default, parameterized, and copy constructor). You might find that multiple parameterized constructors are useful, and you are welcome to add more than one. Note that each copy constructor should allocate new memory for that Image’s pixel buffer ... sharing it across images is problematic (who owns the memory and is responsible for freeing it? ... we aren’t worried about memory leaks yet, but we will soon).

All data members (width, height, data) must be PRIVATE. That said, you may add methods to access data ... your class may return a pointer to your buffer via a method.

You will also need a method to reset the image’s size. This is because the driver program is calling the default constructor, and the size will need to be reset when you want to put a valid image in it. I recommend something like:

void ResetSize(int width, int height);

== 2. Write additional functions to manipulate images ==

HalveInSize: the output image should be half the width and height. Pixel (i, j) in the output should be the same as pixel (2*i, 2*j) in the input.

LeftRightConcatenate: take two input images that have the same height and make a single image where they are combined left-to-right. I.e.: (A) + (B) = (AB)

TopBottomConcatenate: take two input images that have the same width and make a single image where they are combined left-to-right. I.e.: (A) + (B) = (A) 

(B)
Blend: take two input images that have the same width and height and blend them together. If the “factor” is 0.8, then the output image should be 80% image 1 and 20% image 2. (This would mean a “0.8*V1 + 0.2*V2” summation for each channel.) Note: for now, you can assume all inputs are valid. That is, if you are doing a left-right concatenation, then the heights of the two images are the same.

The signatures for all of these functions is in functions.h.

== File structure ==

Your Image class should go in its own file (.h / .C). I have provided a Makefile that compiles your program. You are welcome to change it. Your Makefile should contain only relative paths. (The GTF grader should be able to grade this without modifying your files!)

== How we will test ==

We will test it by running the main program that can be found on the class website. We will also look at your code.

== What to turn in ==

Turn in everything we need to grade ... source code & Makefile. You do not need to turn in the .pnm source files.

== Additional notes ==

Note 1: you do not need to worry about memory leaks yet. That will come in future projects.

Note 2: you now are writing C++ ... make sure you use “.C” (and not “.c”).

Note 3: C++ is touchy about malloc. You will need to cast the return type of malloc. As in:
```
data = (unsigned char *) malloc(3*w*h);
```
where
```
data = malloc(3*w*h);
```
was fine in C

== My advice ==

This project will force you to retrofit your 3A code. I recommend you test frequently. For example, start with a main function like:
#include "image.h"
#include "functions.h"

int main(int argc, char *argv[])
{
    ReadImage(argv[1], img);
    WriteImage(argv[2], img); /* note the original prompt takes img8 */
}

and make sure that it produces an identical output ("diff"). If not, you are just going
to get yourself in deep.