Due Friday, October 27, 11.59pm, Late Deadline, October 29, 11.59pm

In part 2 of this project, you will implement the following methods and augment the Image class:

Image Representation

Your image class should hold the following attributes and implement the associated methods (may include additional attributes/methods as needed).

```
class Image {
   private:
              // image dimensions
      int width, height;
              // pointer to the dynamically allocated image array
      int *image_array;

   public:
      Image();      // constructor - creates an empty image object,
                    // creates an image object by reading the input file
      Image(string input_file);
                    //creates an image object using the given dimensions
      Image (int width, int height);
      ~Image();     // destructor  - provide as many destructors as needed

                    // accessors/mutators
      int getWidth();
      void setWidth(int w);
      int getHeight();
      void setHeight(int h);

                    // set/get an image pixel given row and column addresses
                    // pixel is a 3 element r,g,b triple
      void getImagePixel (int col, int row, int *pixel);
      void setImagePixel (int col, int row, int *pixel);

                    // reads an image from  the given input image file
      read(string infile);
                    // writes image to file
      write(string outfile);

            // converts RGB to grayscale (use R*0.299+G*0.587+0.114*B)
      void toGrayscale();

            // flips horizontally  each row of pixels
      void flipHorizontal()

            // flips the blue component  of the image about 255
      void negateBlue();

            // sets teh red coomponent of each pixel to zero
      void flattenRed();
}
```

Tasks.

You will use the same input image from part 1. You will add the following public methods (see above). **Maintain a temporary copy of the processed image, else it will be difficult to test each method**

sequentially. Write a new version of the getPixel()/setPixel() to receive/send data to the processed image (and any other needed changes to accommodate the new functions).

1. You will implement 4 public functions as follows:

   a) **void toGrayscale().** This method will convert the original RGB PPM image into a grayscale image. You can use the following conversion formula

   $$float\ \ grayVal = R * 0.299 + G * 0.587 + 0.114 * B$$

   where R, G, B are the red, green and blue values of of the pixel. Note that this is a float operation, the resulting value will be in the range 0-255.0, must be converted into an integer, and will be replicated for all 3 components of the pixel.

   b) **void flipHorizontal().** This function will simply flip the pixels in each row horizontally.

   c) **void negateBlue().** This function will flip the blue pixel values around 255(the max value). For instance, if the blue pixel value is 100, then it becomes 255-100 = 155.

2. **Testing.** Write a simple text menu system that asks the user to choose one of the 4 functions. The output of each function can go into a specific output file; name that file in your feedback ("writing to yosemite_1.ppm", for example).

3. **Documentation.** Generate an updated doxygen documentation of your sources; **each of your methods should be documented, including each input parameter, return value, etc.**

Evaluation:

- As per rubric (on Canvas).
- **To Turn in to Canvas:** All source code files, sample images images (from testing as described above) in PPM or PNG format, doxygen based documentation.
- **Late Policy.** Upto 2 days and for a maximum of 50% credit. a reduction of 25% credit, each day. No credit beyond 2 days past the deadline.