CSE 8A Programming Assignment 6

Name should be formatted as (last, first)
If you are working solo you may leave the right column blank.

Name:	Darren Yeung	Name:	
PID:	A15943292	PID:	
Email:	dyeung@ucsd.edu	Email:	

Part 1 Testing Code:

```
// Copy and paste the code from your test methods (including comments!) here
// Make sure to set the font to Courier New
// IMPORTANT: Make sure your code is properly formatted and commented.
// Code that does not have correct indentation and comments will lose marks.
```

GrayScale Test 1

```
public static void testGrayscale() {


      Color[][] test2 = {

        {Color.red},


      };

      Image img2 = new Image(test2);
      img2.explore();

      Image result2 = grayscale(img2);
      result2.explore();


    }
```

GrayScale Test 2

```
public static void testGrayscale() {


        Color max = new Color(255,100,70);

        Color[][] test2 = {

          {max},


        };

        Image img2 = new Image(test2);
        img2.explore();

        Image result2 = grayscale(img2);
        result2.explore();

    }
```

Blend test 1

```
public static void testBlend() {
        // Create two 1x1 2D arrays
        Color[][] test1_1 = {{new Color(0, 0, 0)}};
        Color[][] test1_2 = {{new Color(100, 255, 100)}};

        // Create the test images and call blend
        Image result1 = blend(new Image(test1_1), new Image(test1_2));

        // Explore the image and ensure that the pixel color is (50, 50, 50)
        result1.explore();

        // TODO: Add two more test cases
    }
```

Blend Test 2

```
public static void testBlend() {
```

```
        // Create two 1x1 2D arrays
        Color[][] test1_1 = {{new Color(0, 0, 0)}};
        Color[][] test1_2 = {{new Color(100, 255, 100)}};

        // Create the test images and call blend
        Image result1 = blend(new Image("images/cat.jpg"), new
   Image("images/texture.jpg"));

        // Explore the image and ensure that the pixel color is (50, 50, 50)
        result1.explore();

        // TODO: Add two more test cases
    }
```

Crop test 1:

```
public static void testCrop() {
        // Create a 4x4 2D array
        Color[][] test1 = {
            {Color.black, Color.black, Color.red, Color.red},   // row0
            {Color.black, Color.black, Color.red, Color.red},   // row1
            {Color.black, Color.black, Color.red, Color.red},   // row2
            {Color.red, Color.red, Color.red, Color.red}        // row3
        };

        // Create the test image and call crop
        Image img1 = new Image(test1);
        Image result1 = crop(img1, 1,1);

        // Visualize the result and make sure that it is 3x2 and that all
   colors
        // are black (255, 255, 255) (first three rows and two columns)
        result1.explore();

        // TODO: Add two more test cases
    }
```

Crop test 2:

```
public static void testCrop() {
        // Create a 4x4 2D array
        Color[][] test1 = {
            {Color.black, Color.black, Color.red, Color.red},   // row0
            {Color.black, Color.black, Color.red, Color.red},   // row1
            {Color.black, Color.black, Color.red, Color.red},   // row2
            {Color.red, Color.red, Color.red, Color.red}        // row3
```

```
        };

        // Create the test image and call crop
        Image img1 = new Image("images/cat.jpg");
        Image result1 = crop(img1, 300,300);

        // Visualize the result and make sure that it is 3x2 and that all
colors
        // are black (255, 255, 255) (first three rows and two columns)
        result1.explore();

        // TODO: Add two more test cases
    }
```

Part 1 Explanation:
Briefly explain why you chose each test and how you know your code is working correctly (or not).

GrayScale Test 1: I chose this test because I wanted to test out of my code works with one pixel and it doesn because it changed the single red pixel to a gray one.

GrayScale Test 2: I chose this test because I wanted to see if my code works with a max value of 255 in the red component of the color and it works because I did the grayscale convertor calculation by hand and I got 130 for RGB which was what the code gave me.

Blend test 1: I chose this test because I wanted to see if my code works with both max value of 255 in the green section and it also being one pixel. I know it works because I calculated the average of the two images by hand and the java image program thing gave back the same values for each value of RBG.

Blend test 2: I chose this test because I wanted to see if my code works with an actual full sized image of a cat. I know it works because the resulting image is the same as the blended image that was shown on assignment page.

Crop test 1 : I chose this test because I wanted to see if my code can bring back the first element in the 2d array(row 0 , column 0) and it worked because it returned the first pixel which was black.
Crop test 2: I chose this because I wanted to see if my crop method can work on a full sized cat picture and I know it works because of the fact that it returned an image with 300 pixels wide and 300 pixels in height (which was what I wrote in the code) and it only showed the cat's right ear.

Part 2: Custom Filter

```
// Copy and paste the code from your custom filter method(s) here
// Copy and paste the code from your main method (including comments!) here
// Make sure to set the font to Courier New
// IMPORTANT: Make sure your code is properly formatted and commented.
// Code that does not have correct indentation and comments will lose marks.
```

Method:

```
public static Image negativeImage(Image img){

      Color[][] revised = new Color[img.getHeight()][img.getWidth()];
      Color[][] tobeRevised = img.getPixels2D();

      int red;
      int green;
      int blue;
      int revisedred;
      int revisedgreen;
      int revisedblue;
      for(int row = 0; row < img.getHeight(); row++){

        for(int col = 0; col < img.getWidth(); col++){

          red = (tobeRevised[row][col]).getRed();
          green = (tobeRevised[row][col]).getGreen();
          blue = (tobeRevised[row][col]).getBlue();

          revisedred = 255 - red;
          revisedgreen = 255 - green;
          revisedblue = 255 - blue;
          revised[row][col] = new
   Color((revisedred),(revisedgreen),(revisedblue));
        }
    }

    Image revisedd = new Image(revised);
    return revisedd;

  }
```

Main method:

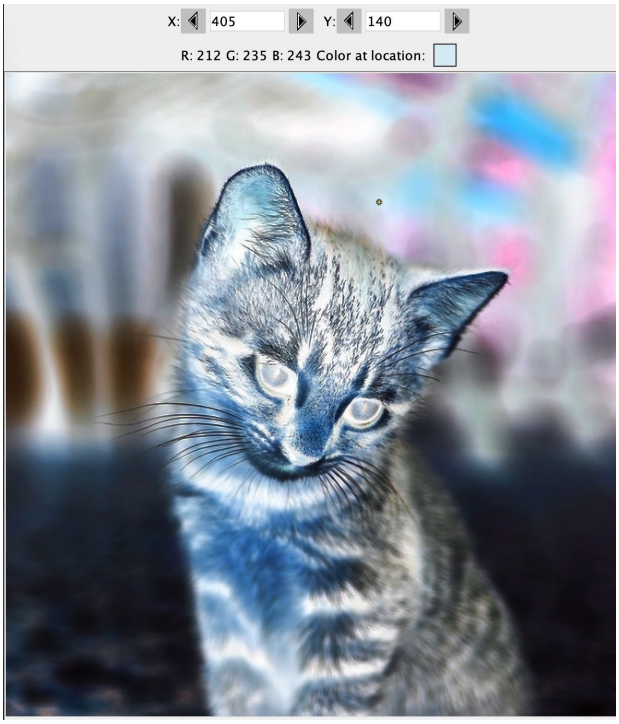
```
public static void main(String[] args) {
        // You may want to uncomment one test at a time

        //testGrayscale();
        //testBlend();
        //testCrop();
        testNegative();


        // TODO: Add code for Part 2 here
    }
```

Part 2 Before/After:

Show us the before and after of running your filter on the input image(s). Make sure that these images are the ones produced by your main method as shown above. Also describe what you were trying to achieve with your filter(s) and if you did not quite achieve that, what were the challenges you ran into.

Before:

After:



I wanted to make a negative image of an original image.To do this, I had to subtract the original color components from 255 and the answer I got was the new color component of the new pixel. The only problem I ran into was using getweight instead of getwidth as I got confused with height and weight and width. It took around 10 minutes to see what I did wrong but I fixed it at the end.

Known Bugs or Issues:
If you have known bugs or issues with your code, let us know here. If you think it works correctly, justify why.

No known bugs or issues with my code. My code works correctly because for every method, I have tested it by hand to see if the color components matches up with what it is supposed to be. My methods do indeed work as they produce the same result I got by hand.