Activity 1 – How to Get the Negative of an Image **Name_____**

1. Find a digital image of your own that you would like to work with and save it to the folder that you will be working from. Ideally, the image should be in the jpeg format, but others will work.

2. Launch the Octave GUI.

3. Set the working directory to your own personal folder.

4. Load the image toolkit.

5. Clear the screen (just to make it easier to focus on your work).

6. The image can be read into Octave using imread. This will convert the image into a matrix, which we want to store as a variable. The variable can be named anything you'd like, but it is ideal to choose a name that will be descriptive.

```
myImage = imread('sun.jpg');
```

*** Notice that a semicolon is used at the end of the line. The semicolon suppresses the output of a command, meaning that the values will not print to the screen. With images this is very useful because they are typically very large. Try it once without the semicolon for fun to see what happens. The semicolon will not affect the values being stored.***

7. Let's investigate some aspects of the image.
   a. *What are the dimensions of your image?*

   b. *What does each dimension represent?*

   c. *What are the values at (1, 1, 1), (1, 1, 2), and (1, 1, 3)?*

   d. *Follow this link and plug in the values you just found. What does this color represent? Does it fit with the context of the image?*

8. The pixel values are currently represented in the form of uint8 (unsigned integer of 8 bits) from 0 to 255. It will be helpful for us to deal with these as doubles from 0 to 1, using im2double.

```
myImage = im2double(myImage);
```

Repeat step 7a. *What is the relationship between the current values and the ones you found before the image was converted?*

9. We will now use the command rgb2gray to convert the image to grayscale. This will make finding the negative more meaningful. By making a new variable name, we can keep the values from the original image.

```
myGrayImage = rgb2gray(myImage);
```

What are the new dimensions of the image?

What is the value of pixel (1, 1)?

10. To obtain the negative of an image, all intensity values are inverted. That means that the amount of light that was originally present will now be restricted. Since the pixels are now scaled from 0 to 1, we can accomplish this by subtracting all values from 1.

    Let's make a new variable for the negative, for example myNegImage. This will allow us to keep the values of the original grayscale image and compare it to the negative.

    ```
    myNegImage = 1 – myGrayImage;
    ```

    What are the dimensions of the negative image?

11. Let's investigate a little. Take a 5 by 5 sample from the top-left of the original grayscale, and then do the same for the negative.

    ```
    myGrayImage(1:5,1:5)
    myNegImage(1:5,1:5)
    ```

    Notice that we did not use a semicolon here since we want to see the values and not suppress them.

    Are the values what you expected?

12. Now we can take a look at our image using the imshow command. Images and graphs are displayed in what are referred to as figures in Octave. For us to view the original, grayscale, and negative, we will make a figure to display each.

    ```
    figure(1)
    imshow(myImage)
    figure(2)
    imshow(myGrayImage)
    figure(3)
    imshow(myNegImage)
    ```

    Can you tell what taking the negative accomplished?

13. Now that you have successfully processed an image, you can save the result using imwrite. This command takes three parameters; the image variable name, the name to save it as (use ' '), and the format, (also use ' ').

    ```
    imwrite(myNegImage, 'NegativeOfSun.jpg','jpg')
    ```

    It is recommended for our purposes to use jpeg as the file format. Other formats are supported, but could result in more complicated issues.