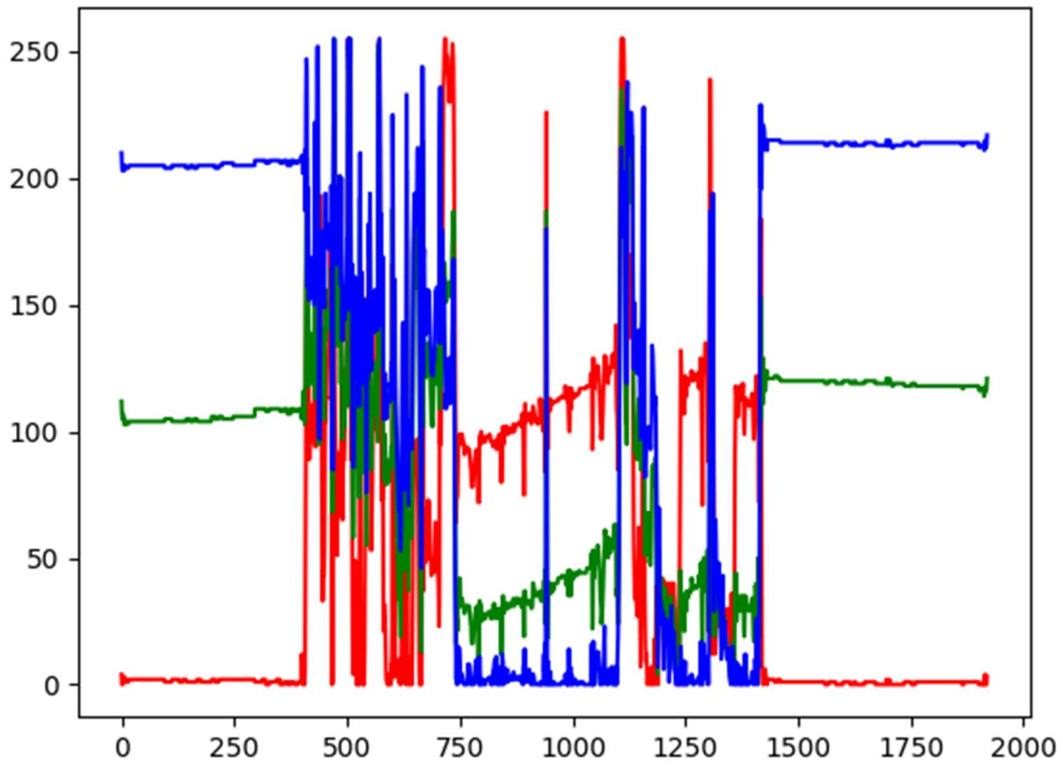
HOMEWORK- 0

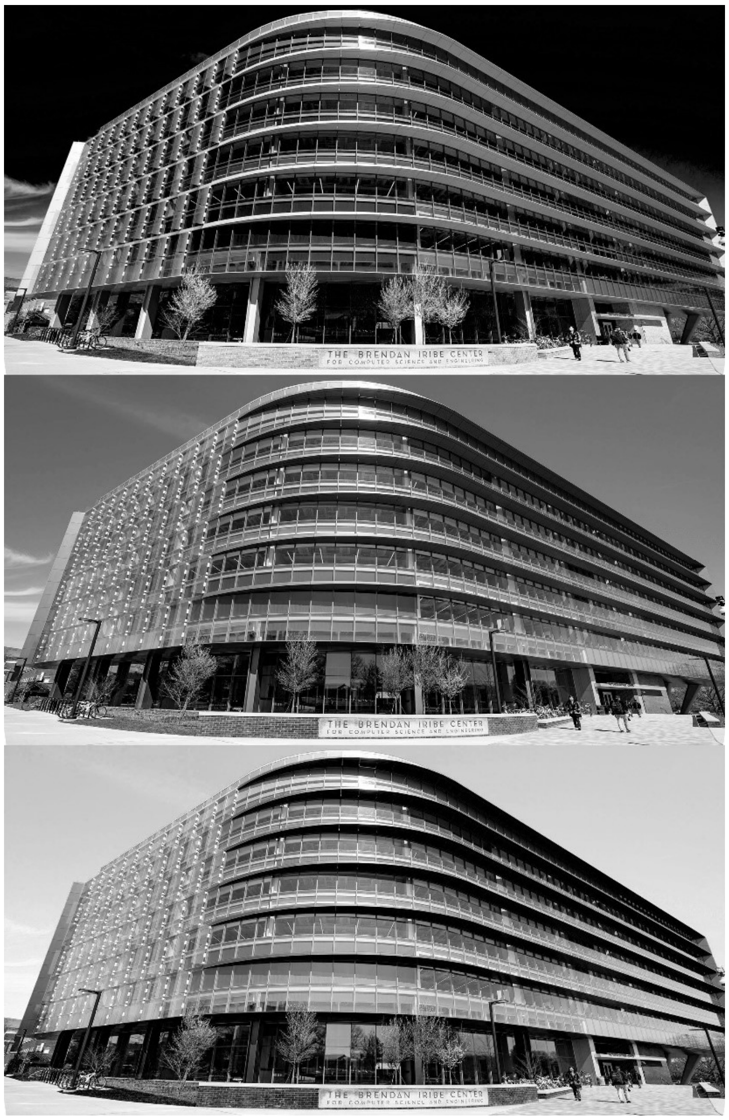
1.) Plot the R, G, B values along the scanline on the 250th row of the image.

```
#This function plots the required row on the scanline
def scanLine(imgArray,row):
    temp = imgArray[250,:]
    print(temp.shape)
    r, g, b = list(imgArray[250,:,0]),
list(imgArray[250,:,1]),list(imgArray[250,:,2])
    plt.plot(r,'-',color='red')
    plt.plot(g,'-',color='green')
    plt.plot(b,'-',color='blue')
    plt.savefig('Resources/1_scanline.png')
    plt.show()
```



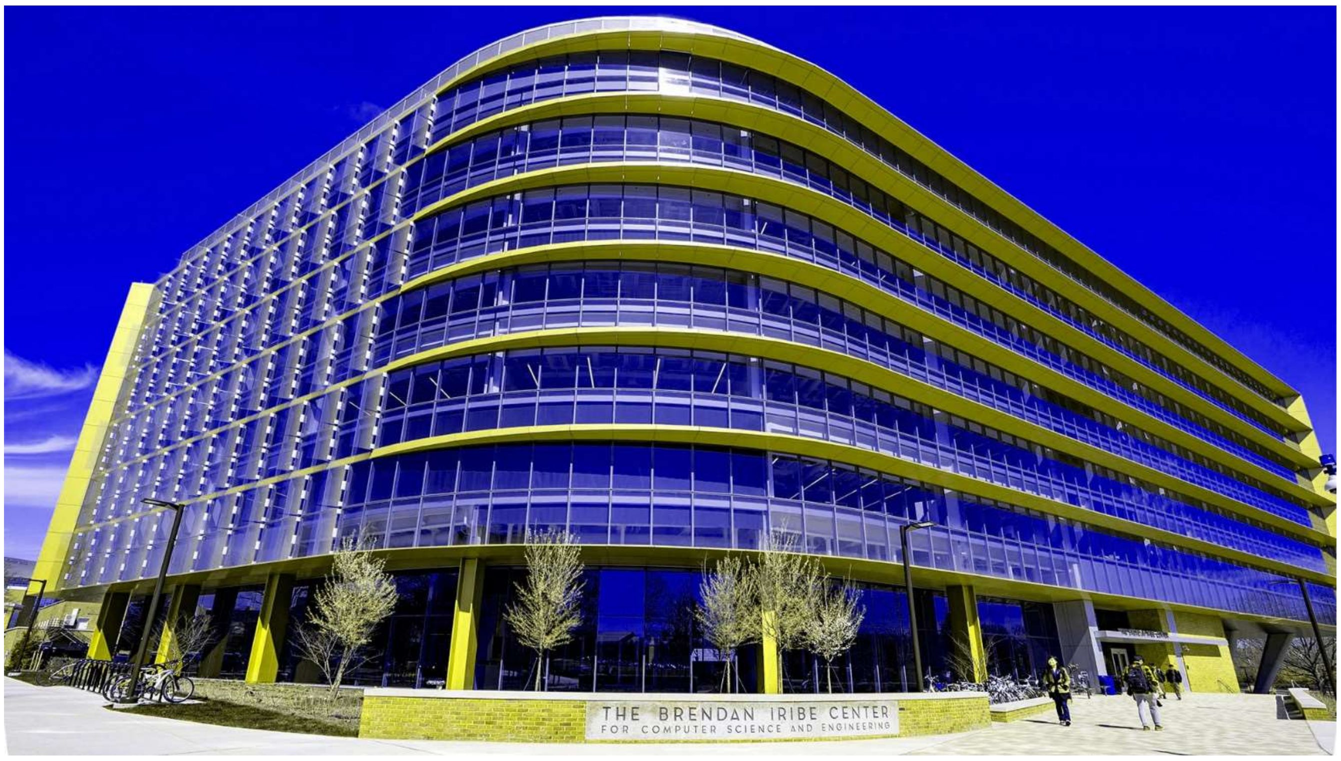
2.) Stack the R, G, B channels of the image vertically.

```
#To stack the three color channels vertically
def verticalStack(imgArray):
    r_channel = imgArray[:,:,0]
    g_channel = imgArray[:,:,1]
    b_channel = imgArray[:,:,2]

    final_img_arr = np.vstack((r_channel,g_channel,b_channel))
    final_img = arrayToImage(final_img_arr)
    final_img.save("Resources/2_concat.png")
```



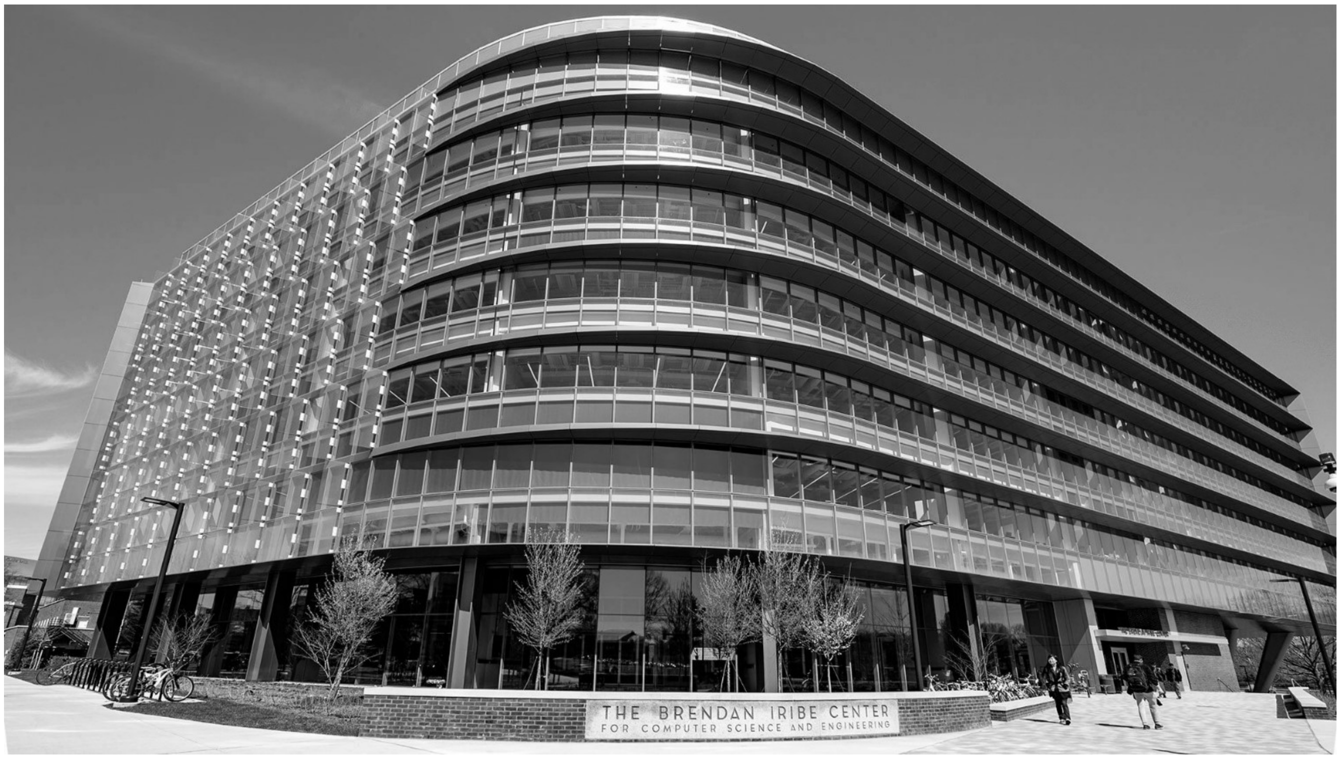
3.) Load the input color image and swap its red and green color channels.

```
#Swapping the color channels red to green and vice versa.
def swapChannels(imgArray):
    temp = imgArray[:,:,1]
    imgArray[:,:,1] = imgArray[:,:,0]
    imgArray[:,:,0] = temp

    final_img = arrayToImage(imgArray)
    final_img.save("Resources/3_swapchannel.png")
```



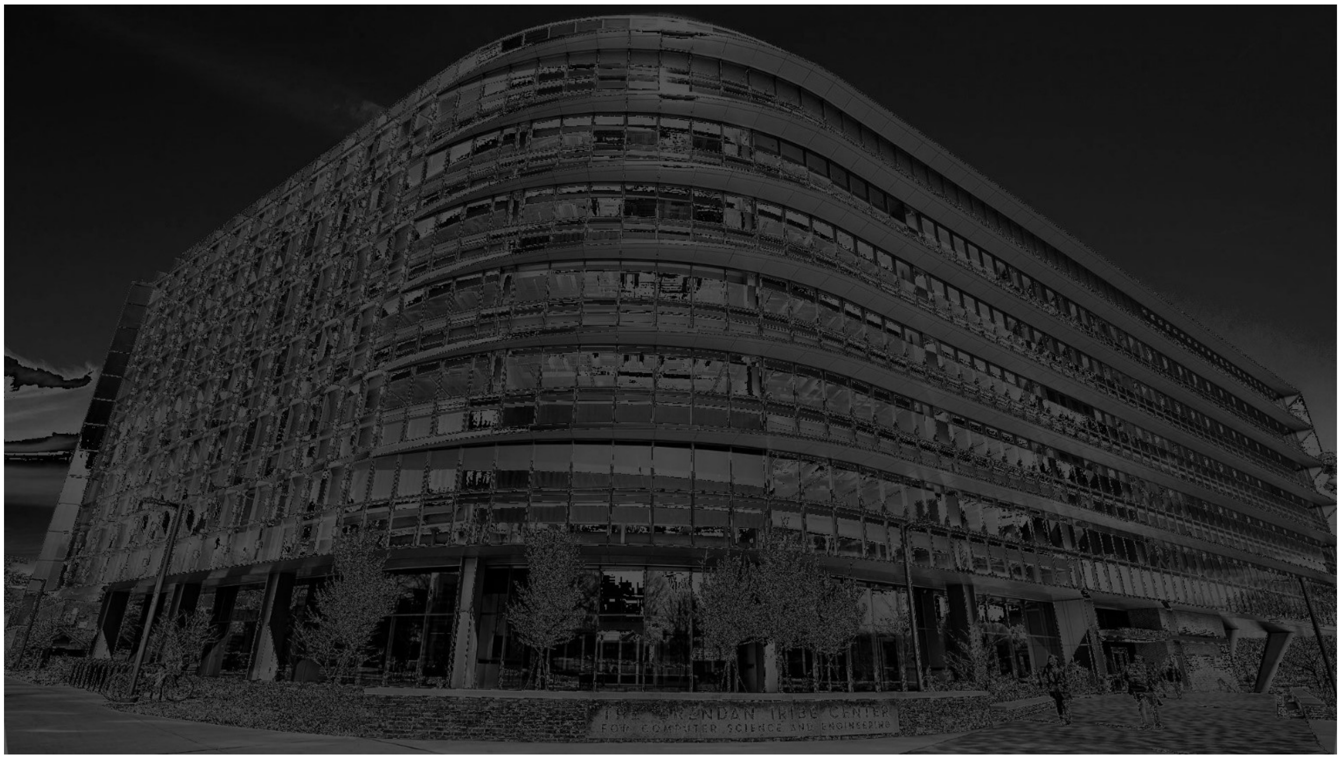
4.) Convert the input color image to a grayscale image.

```
# RGB to Grayscale conversion
def rgbToGrayscale(imgArray):
    r_channel = imgArray[:,:,0]
    g_channel = imgArray[:,:,1]
    b_channel = imgArray[:,:,2]

    Grayscale_img = b_channel* 0.2126 + g_channel * 0.7152 + r_channel *
0.0722
    # print(Average_Gray)
    final_img = arrayToImage(np.uint8(Grayscale_img))
    final_img.save("Resources/4_grayscale.png")
```



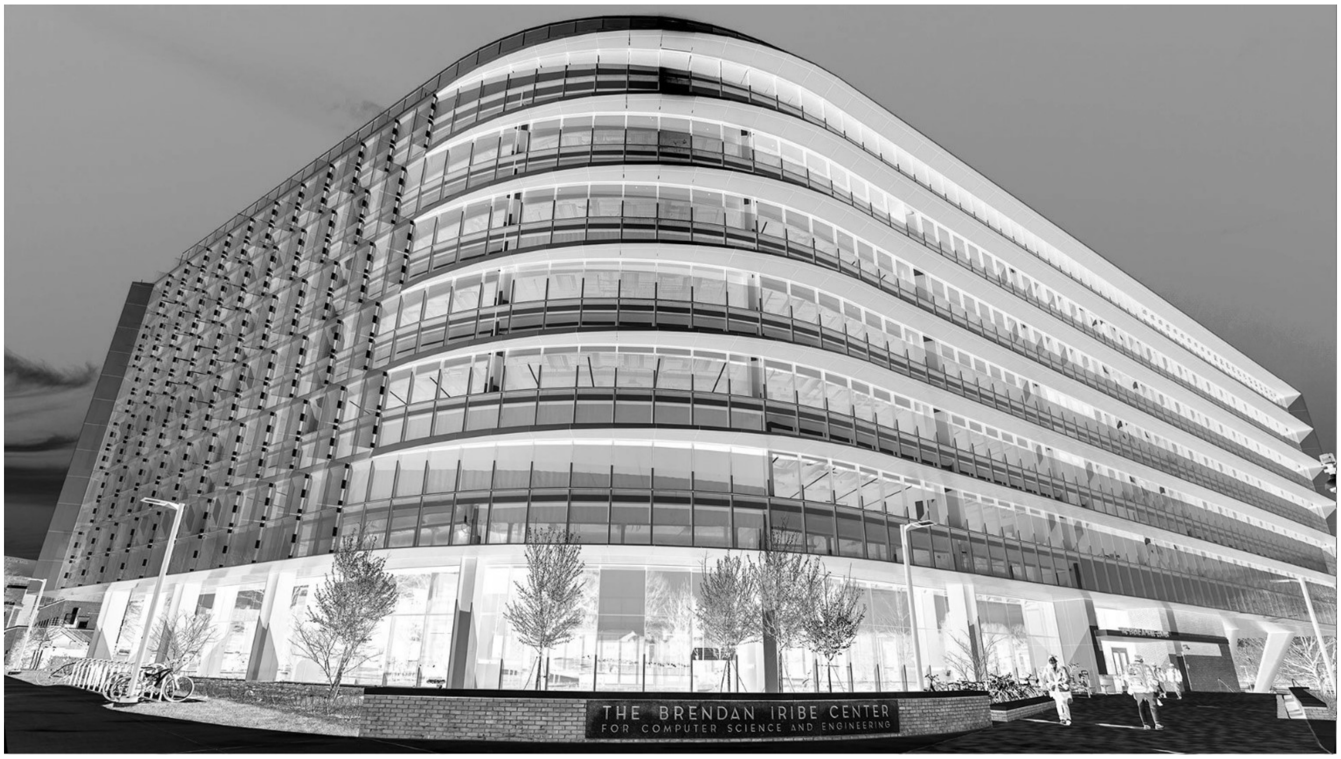
5.) Take the R, G, B channels of the image. Compute an average over the three channels. Note that you may need to do the necessary typecasting (uint8 and double) to avoid overflow.

```
# Average method
def average(imgArray):
    r_channel = imgArray[:,:,0]
    g_channel = imgArray[:,:,1]
    b_channel = imgArray[:,:,2]
    Average_Gray = (b_channel + g_channel + r_channel)/3
    # print(Average_Gray)
    final_img = arrayToImage(Average_Gray)
    final_img.save("Resources/5_grayscale.png")
```



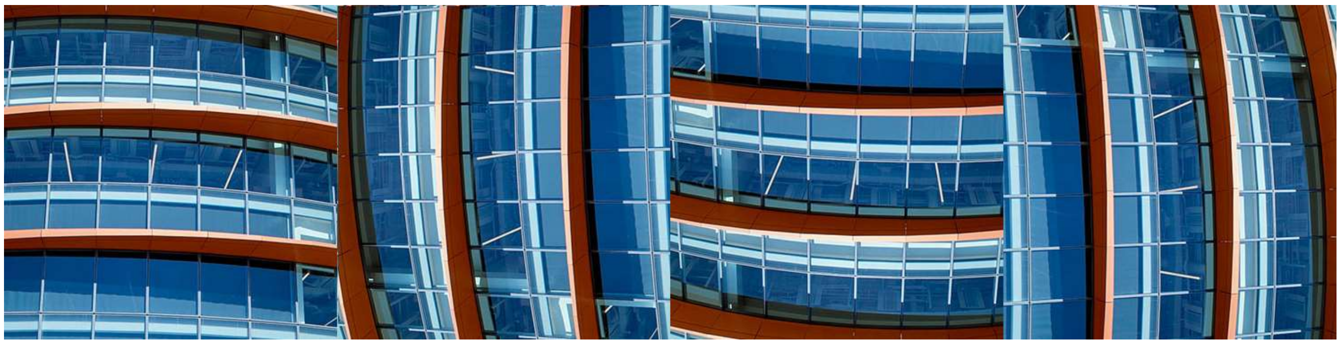
6.) Take the grayscale image in (4), obtain the negative image (i.e., mapping 255 to 0 and 0 to 255).

```python
# Negative image converter from grayscale
def negativeImageConverter(imgArray):
    h,w,_ = imgArray.shape
    r_channel = imgArray[:,:,0]
    g_channel = imgArray[:,:,1]
    b_channel = imgArray[:,:,2]

    Average_Gray = b_channel//3 + g_channel//3 + r_channel//3
    for i in range(0,h-1):
        for j in range(0,w-1):

            Average_Gray[i,j] = 255 - Average_Gray[i,j]

    final_img = arrayToImage(Average_Gray)
    final_img.save("Resources/6_negative.png")
```



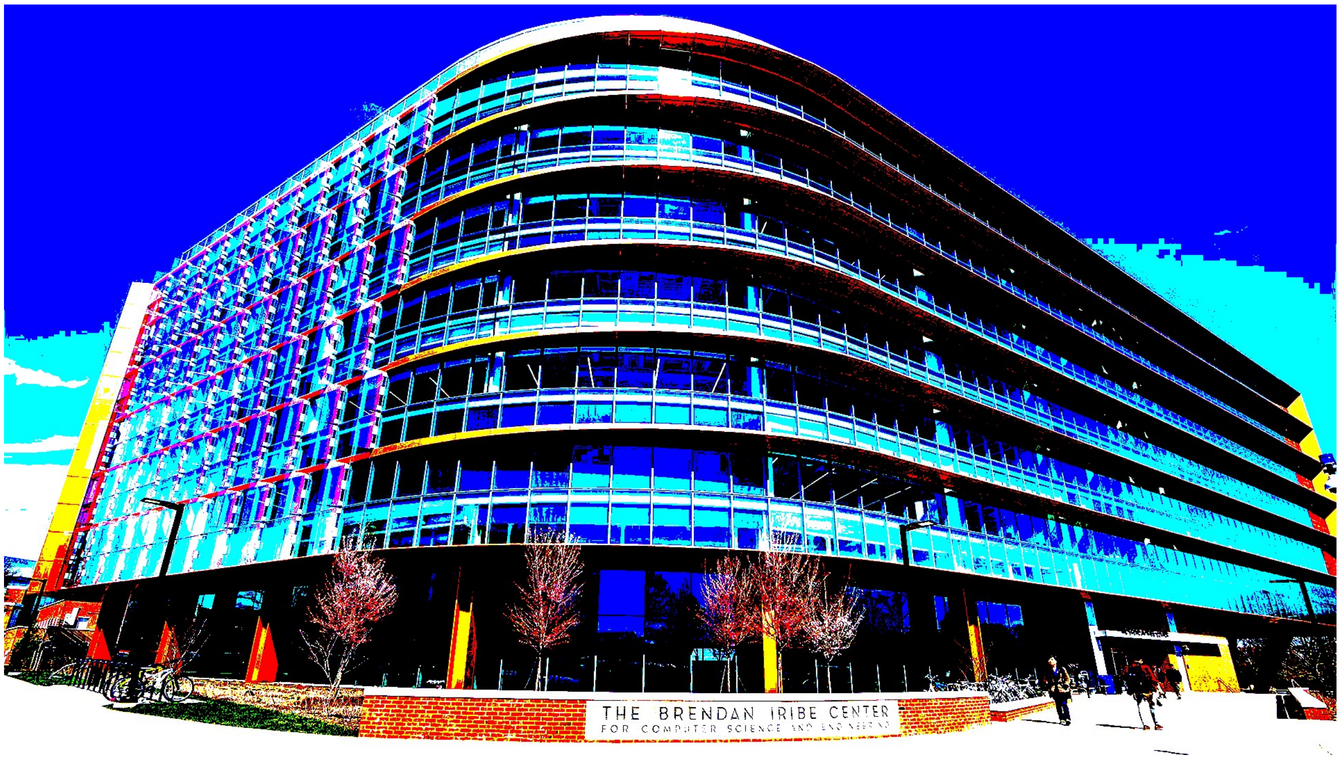
7.) First, crop the original image into a squared image of size 372 x 372. Then, rotate the image by 90, 180, and 270 degrees and stack the four images (0, 90, 180, 270 degrees) horizontally.

```
# Cropping a square image, rotating it and stacking it horizontally.
def cropRotateStack(imgArray):
    squaredCropImg = imgArray[354:726,774:1146]

    rows,cols,channels = squaredCropImg.shape
    new_img_1 = np.zeros([rows,cols,channels],dtype = np.uint8)
    new_img_2 = np.zeros([rows,cols,channels],dtype = np.uint8)
    new_img_3 = np.zeros([rows,cols,channels],dtype = np.uint8)

    #Rotating it by 90degrees left
    for i in range(rows):
        for j in range(cols):
            new_img_1[i,j] = squaredCropImg[j-1,i-1]
            new_img_1 = new_img_1[0:rows,0:cols]

    #Rotating it by 180 degrees
    for i in range(rows):
        for j in range(cols):
            new_img_2[i,j] = squaredCropImg[rows - i-1,cols - j-1]
            new_img_2 = new_img_2[0:rows,0:cols]


    #Rotating it by 270 degrees left
    for i in range(rows):
        for j in range(cols):
            new_img_3[i,j] = squaredCropImg[rows - j-1,cols - i-1]
            new_img_3 = new_img_3[0:rows,0:cols]


    final_img_arr =
np.hstack((squaredCropImg,new_img_1,new_img_2,new_img_3))
    final_img = arrayToImage(final_img_arr)
    final_img.save("Resources/7_rotation.png")
```



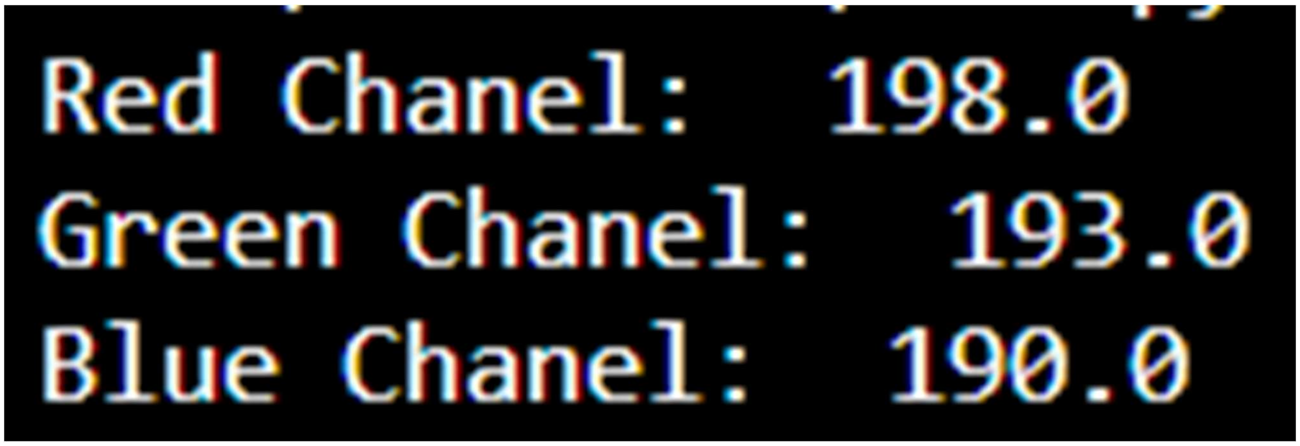
8.) Create another image with the same size as the image. First, initialize this image as zero everywhere. Then, for each channel, set the pixel values as 255 when the corresponding pixel values in the image are greater than 127.

```python
#Creating a mask according to the given criteria
def imgMask(imgArray):
    rows,cols,channels = imgArray.shape
    new_img = np.zeros([rows,cols,channels],dtype = np.uint8)

    new_img[np.where(imgArray>127)] = 255
    final_img = arrayToImage(new_img)
    final_img.save("Resources/8_mask.png")
```



9.) Report the mean R, G, B values for those pixels marked by the mask in (8).

```
def maskRGBValues(imgArray):
    rows,cols,channels = imgArray.shape
    temp_array = np.where(imgArray>127)

    red_arr =imgArray[temp_array][0]
    green_arr =imgArray[temp_array][1]
    blue_arr =imgArray[temp_array][2]
    print("Red Chanel: ", np.mean(red_arr))
    print("Green Chanel: ", np.mean(green_arr))
    print("Blue Chanel: ", np.mean(blue_arr))
```



```
Red Chanel:  198.0
Green Chanel:  193.0
Blue Chanel:  190.0
```

10.) Take the grayscale image in (3). Create and initialize another image as all zeros. For each 5 x 5 window in the grayscale image, find out the maximum value and set the pixels with the maximum value in the 5x5 window as 255 in the new image.

```python
def nonMax(imgArray):
    r_channel = imgArray[:,:,0]
    g_channel = imgArray[:,:,1]
    b_channel = imgArray[:,:,2]
    grayscale = (r_channel + g_channel + b_channel)//3
    # average = int(average)
    rows,cols = grayscale.shape
    new_img = np.zeros([rows,cols],dtype = np.uint8)

    for i in range(0,rows-4):
        for j in range(0,cols-4):
            temp = grayscale[i:i+5,j:j+5]
            max = np.amax(temp)
            index = np.where(temp == max)
            new_img[i:i+5,j:j+5][index] = 255




    final_img = arrayToImage(new_img)
    final_img.save("Resources/10_nonmax.png")
```