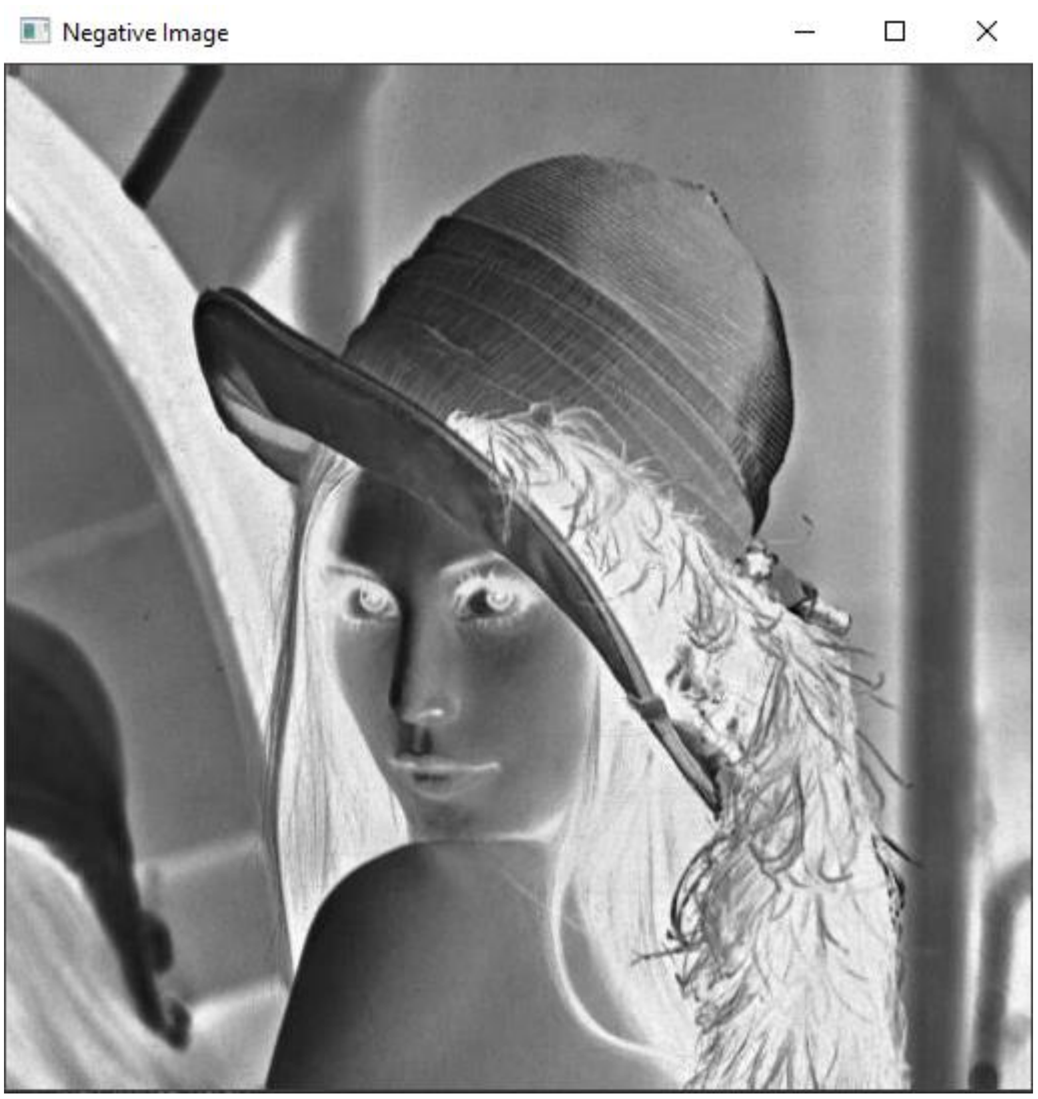
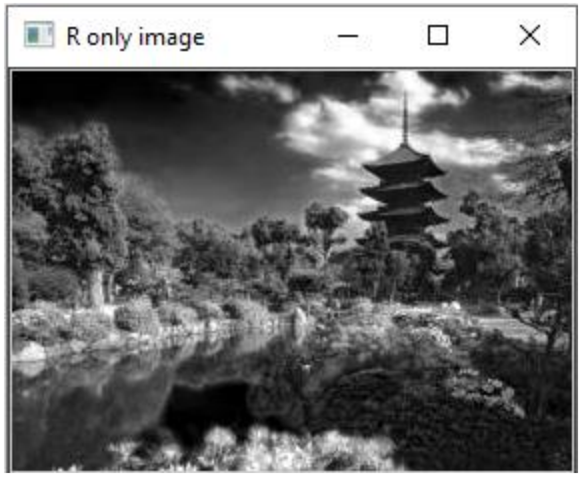
Assignment #3 – EE 568 – Digital Image Processing – Winter 2021

Name: Kalana Sahabandu

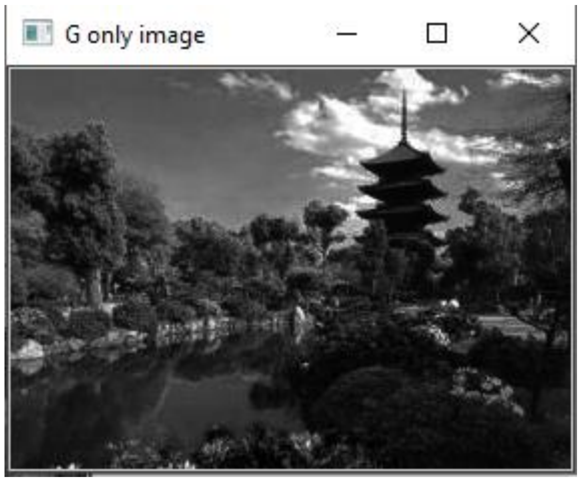
Question 1



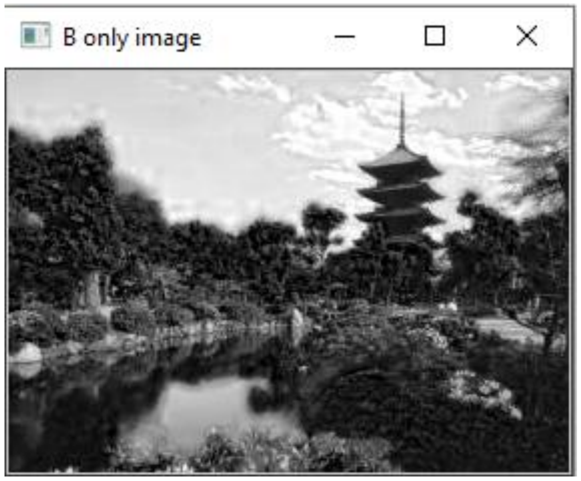
Question 2



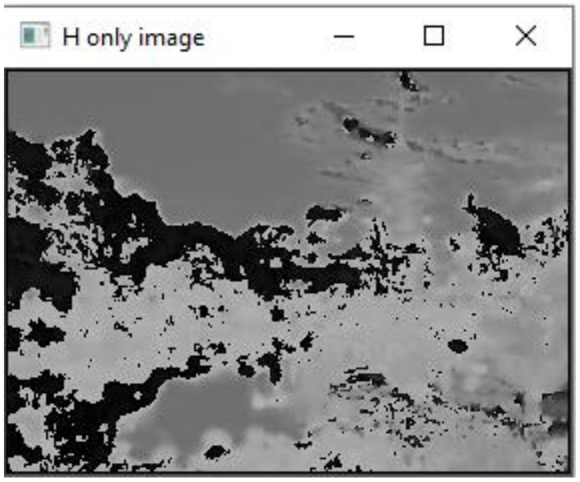
R image



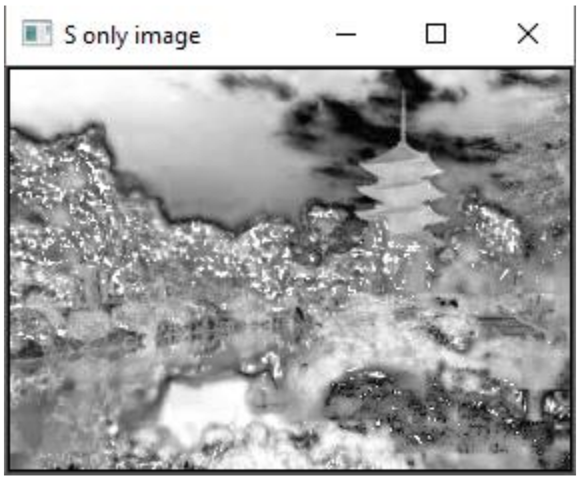
G image



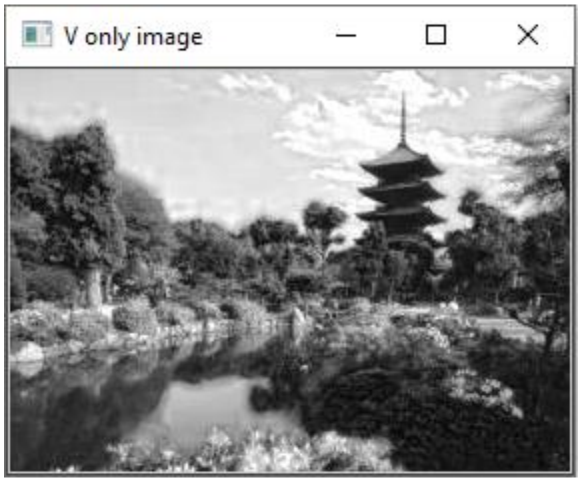
B image



H image



S image

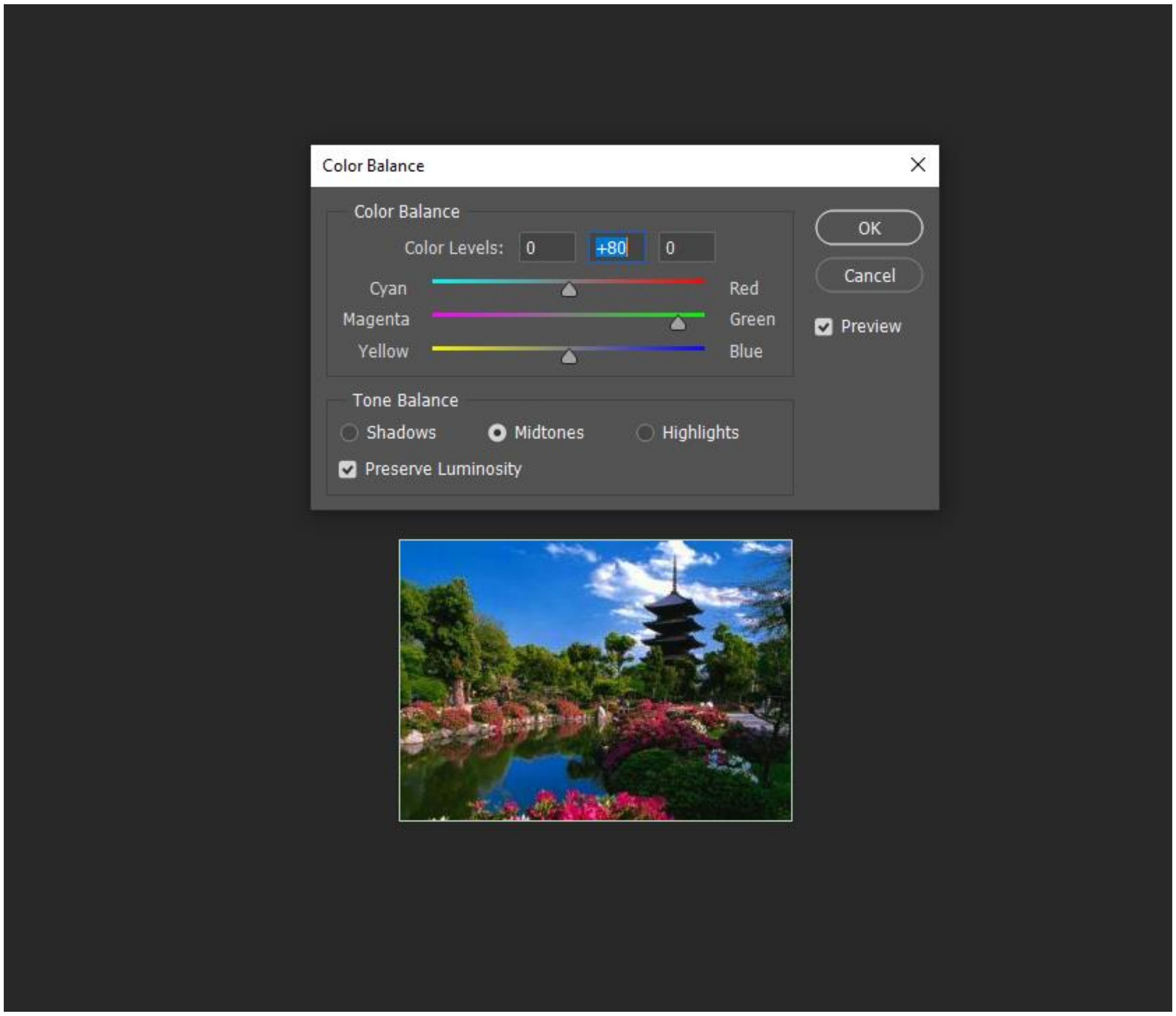


V image

Question 3

<u>Method 1: Reducing Magenta (R and B) values and increasing Green values</u>

After playing around with the image colors in Photoshop I have found that intensity of the magenta color is too high while the intensity of the green channel is low resulting in magenta tint in the original image



Color Balance

Color Levels: 0 +80 0

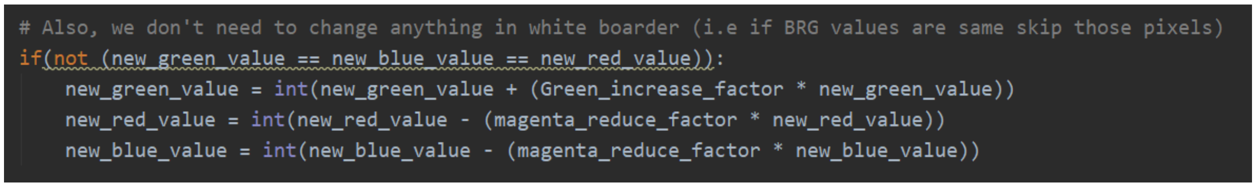
Tone Balance: Shadows [✓ Midtones] Highlights

[✓] Preserve Luminosity

[✓] Preview

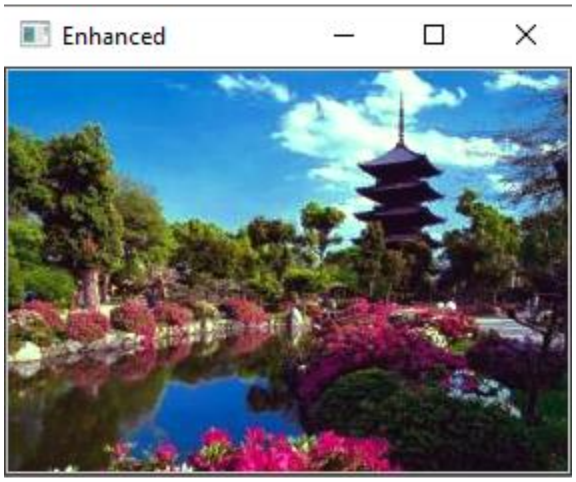
Increasing green intensity while reducing magenta (red + blue) intensity using photoshop

Once I have figured out, I need to reduce red and blue intensity and increase the green intensity of the image, using python I started processing each pixel value to reducing red and blue while increasing green value. After playing around with by how much I need to decrease and increase above colors, I have finally settled on increasing green value of each pixel by 0.6 (60%) and decreasing red and blue values of each pixel by 0.01 (1%). However, this resulted in turning the white boarder around the image to look quite green since the value of green is increased in the pixels related to the boarder section. Therefore, I have added following conditional statement to skip color corrections in white boarder



```
# Also, we don't need to change anything in white boarder (i.e if BRG values are same skip those pixels)
if(not (new_green_value == new_blue_value == new_red_value)):
    new_green_value = int(new_green_value + (Green_increase_factor * new_green_value))
    new_red_value = int(new_red_value - (magenta_reduce_factor * new_red_value))
    new_blue_value = int(new_blue_value - (magenta_reduce_factor * new_blue_value))
```

Also, while coding this section I have realized that increasing green value of each pixel by 0.6 could result in values greater than 255 which could result in errors, therefore I have added following conditional statement to make sure to force green value to be 255 in case newly calculated green value is greater than 255 (which could result in uint8 overflow).



Enhanced image using Method 1

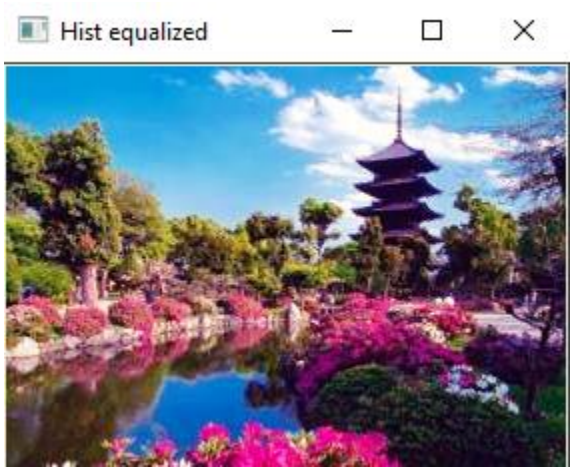
Method 2: Histogram Equalization

While I was looking into more ways to make images look more natural, I came across following article on Wikipedia

https://en.wikipedia.org/wiki/Histogram_equalization

That helps to adjust intensities of colors to be better distributed using histogram equalization such that we could remove magenta tint from the image.

As per my first step, I have split RGB channels from the Image data matrix and feed it to a function called histogram_equalization() where I calculate the intensity frequency of each pixel in the channel then I have calculated cumulative sum and using that information I have normalized each pixel information of the channel (I have performed normalization for all RGB channels). Once normalization for each channel is done, I have reconstructed the image by merging all the 3 channels together such that I can show the resulting image using cv2.imshow() method.



Enhanced image using Method 2

Out of these two methods, I prefer method 2 over method 1 since method 1 is more specific to the given image while method 2 can be used to color correct any image. In addition, resulting image from method 1 looks quite dark compared to the resulting image I got from method 2 which made the resulting image from method 2 looks more natural compared to the resulting image from method 1. Therefore, I believe method 2 is much more sophisticated and yields better results (more natural looking images) compared to method 1.

Note: Code for question 1, 2 and 3 can be found in the .zip file I have attached with the submission where Q1, Q2 and Q3 directories represent above questions, respectively. Both method 1 and method 2 of Question 3 is implemented as functions within Q3 class of the Q3.py python source file