Digital image processing project.

Supervised by:

- Dr.Zaher Salah.

Assignment 1:

In the first cell we call the libraries (NumPy for working with arrays, PIL to deal with images) that we are going to use in the project :

```
In [1]: from PIL import Image
        import numpy as np
```

1.1 To read and display the images, we wrote this block of code:

```
In [2]: lena_img = Image.open(r"LenaGray.jpg")
        peppers_img = Image.open(r"PeppersGrey.jpg")
```

```
In [3]: lena_img.show()
        peppers_img.show()
```

The results:

2.2 In the following code we taked the top half of the first image and the lower half of the second image and then stored them in image (j) which is image of type 'L' (8-bit pixels) with 256 height and width:

```
In [4]: image_J = Image.new('L' , (256,256))
```

Then, we used the getpixel() function that is used to retrieve the color value of a specific pixel in an image , with its help we can obtain information about the color of the selected pixel .

The putpixel() function is used to set the color value of specific pixel in

```
In [5]: for y in range(256):
            for x in range(256):
                if y <= 128:
                    image_J.putpixel((x,y),lena_img.getpixel((x,y)))
                else:
                    image_J.putpixel((x,y),peppers_img.getpixel((x,y)))
```

the image ,enabling us to change the color of specific pixel by specifying the coordinates and the value of the new color :

To display (j) image:

```
In [6]: image_J.show()
```

The result:

1.3 Create a new 256 x 256 image(K) to making swap:

```
In [7]: image_K = Image.new('L' , (256,256))
```

We used in the following block the Crop() function to cut the image into two parts.the coordinates for the part to be cut are determined by the four digits (column prefix px , row prefix px , column final px , row final px) :

```
In [8]: upper_half=image_J.crop((0,0,256,128))
        lower_half=image_J.crop((0,128,256,256))
```

These two lines use the K image paste() function - which used to combine the two images together - to paste the lower half and upper half of the J image into the K image.

```
In [9]: image_K.paste(lower_half,(0,0))
        image_K.paste(upper_half,(0,128))
```

Then to show the final result:

```
In [10]: image_K.show()
```

Assignment 2:

2.1 Load the images :

```
In [2]: lena_noise = Image.open(r"LenaGrayNoisy.jpg")
        peppers_noise = Image.open(r"PeppersGreyNoisy.jpg")
```

Create two new images :

```
In [3]: new_lena = Image.new('L',(256,256))
        new_peppers = Image.new('L',(256,256))
```

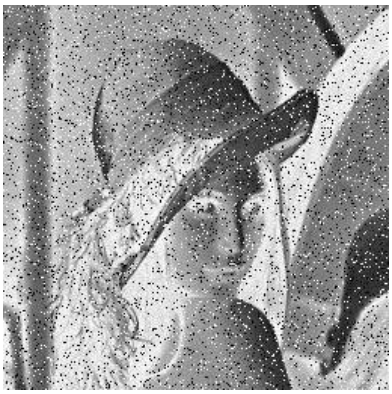
We applied here Negative for lana image:

```
In [4]: for y in range(256):
            for x in range(256):
                new_lena.putpixel((x,y),255-lena_noise.getpixel((x,y)))
```

When performing the image negative, we use the maximum color channel values (in this gray case), which are 255 in the case of 8-bits per pixel images. The process of negativity simply means switching the previous values with opposite ones. For example, if the value of a pixel in the image is 100, after performing the negative operation it will become 255-100 = 155.

To show the result:

```
In [5]: new_lena.show()
```



2.2

To create gray scale Median Filter We used the window() function, it is a function that returns a list containing the coordinates of the pixels adjacent to a given point in the image, based on the coordinates() given to the function. The function takes two variables x and y representing the coordinates of a particular pixel in the image. It starts by checking whether the pixel (is located on the top, bottom right, or left edge of the image. If a pixel is located in either of these states, the neighbouring pixel is reached in that case. If the pixel is not on the edges, the function returns a list containing the coordinates of the nine adjacent pixels for pixel) this list includes the coordinates of the pixel itself and eight pixels the function returns the value, this indicates that it can not be another about it.

```
In [6]: def window3 (x,y):
            if x==0 or y==0 or x==256 or y==256:
                return 0
            return [(x-1, y-1),(x, y-1),(x+1, y-1),(x-1, y),(x, y),(x+1, y),(x-1, y+1),(x, y+1),(x+1, y+1)]
```

Here the window3 function is used to get the coordinates of the pixels adjacent to the current pixel. Then lena_noisy.getpixel(i) is used to get the pixel values for each pixel in the adjacent list. The internal cycle of this iteration estimates the cycle for each pixel in the image and then the average value is calculated using np.median(window_vals) for adjacent pixels, and this average value is applied to the primary pixel (new_lena) in the resulting image. If any exception occurs while getting adjacent pixels, the value of the resulting pixel is set to 0. This program performs the process of applying the median filter to Photo is LenaGrayNoisy.jpg:

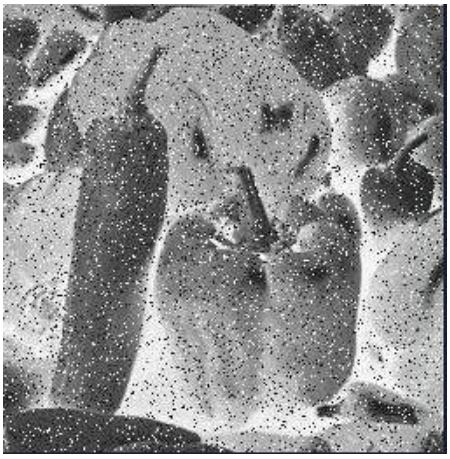
In [7]:
```
for y in range(256):
    for x in range(256):
        try:
            window_val= [lena_noise.getpixel(i) for i in window3(x,y)]
            new_lena.putpixel((x,y) ,int(np.median(window_val)))
        except:
            new_lena.putpixel((x,y),0)
```

In [8]:
```
new_lena.show()
```

And the same steps we did to the pepper image, the codes and results:

In [9]:
```
for y in range(256):
    for x in range(256):
         new_peppers.putpixel((x,y),255-peppers_noise.getpixel((x,y)))
```

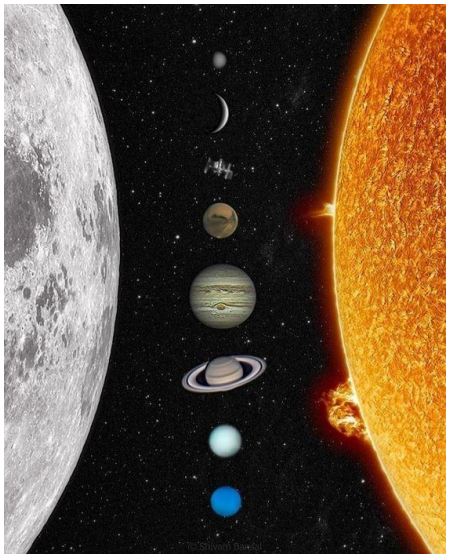
In [10]:
```
new_peppers.show()
```



In [11]:
```
for y in range(256):
    for x in range(256):
        try:
            window_val= [peppers_noise.getpixel(i) for i in window3(x,y)]
            new_peppers.putpixel((x,y) ,int(np.median(window_val)))
        except:
            new_peppers.putpixel((x,y),0)
```

In [12]:
```
new_peppers.show()
```

Extra Examples:
Galaxy Image:

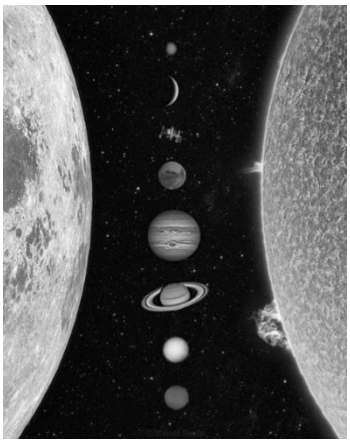


We created an object of the (ImageEnhance.Color) class using the previously opened image galaxy, black_galaxy = im_en.enhance(0), This line enhances the color of the image using the enhance method, black_galaxy should be a black-and-white version of the original "galaxy.jpg" image, Black galaxy image:

```
galaxy = Image.open("galaxy.jpg")

# Creating object of Color class
im_en = ImageEnhance.Color(galaxy)

# showing resultant image
black_galaxy=im_en.enhance(0)
```



Then, we created in the following code a new RGB image with dimensions 564x701 pixels. The image is initialized with all pixels set to (0, 0, 0), which is black.

Inside the loop, there's an if condition that checks whether the current x-coordinate is between 193 and 398, the current pixel is within a specified range in the x-direction. In this case, it copies the pixel value from the corresponding location in the black_galaxy

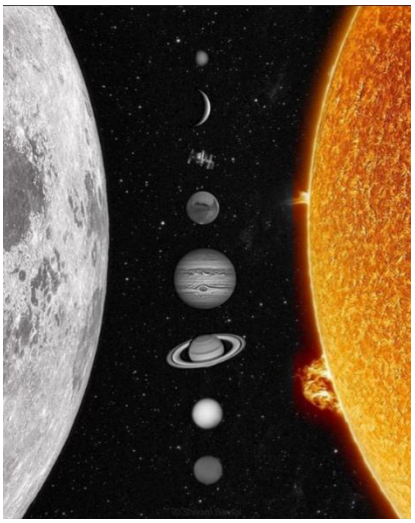
image, otherwise it copies the pixel value from the corresponding location in the galaxy image, and we noticed this interesting result (Color_img1):

```
color_img1 = Image.new( mode = "RGB", size = (564, 701) )

for y in range (701):
    for x in range(564):
        if x>=193 and x<= 398:
            scale_value=black_galaxy.getpixel((x,y))
            color_img1.putpixel((x,y), scale_value)
        else:
            scale_value=galaxy.getpixel((x,y))
            color_img1.putpixel((x,y), scale_value)

color_img1.show()
```

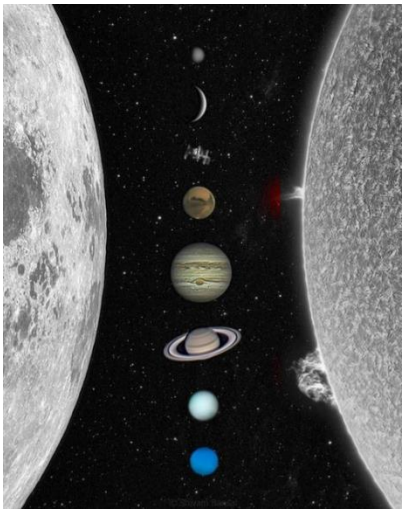
The result:



Then we tried to reversal of images inside the loop, as it shown in the following code, and we noticed this result:

```
color_img2 = Image.new( mode = "RGB", size = (564, 701) )
for y in range (701):
    for x in range(564):
        if x>=193 and x<= 388:
            scale_value=galaxy.getpixel((x,y))
            color_img2.putpixel((x,y), scale_value)
        else:
            scale_value=black_galaxy.getpixel((x,y))
            color_img2.putpixel((x,y), scale_value)

color_img2.show()
```



IN this example,

We tried to make the following image darker than what it is then reverse it.

The original image:

After editing:

we applied the following blocks of code:

```
sun = Image.open("sun.jpeg")

# Create an ImageEnhance object
enhancer = ImageEnhance.Brightness(sun)

# Darken the image using the factor (0.0 means completely black)
darkened_img = enhancer.enhance(0.3)
# Save the darkened image
darkened_img.save('dark_sun.jpeg')
```

```
d_s=Image.open('dark_sun.jpeg')
color_img3 = Image.new( mode = "RGB", size = (736, 1228) )

for y in range (1228):
    for x in range(736):
            scale_value=d_s.getpixel((735-x,y))
            color_img3.putpixel((x,y), scale_value)

color_img3.show()
color_img3.save('dark_sun.jpeg')
```