Digital Image Processing Lab

Assignment by MTH Junaidi AP17110010074 of CSE B

Basic Operations and Intensity Transformation (Point Operations)

This is the first lab exercise and we are to achieve the following

1. Perform the following operations using library functions </br> a. Read, Display and write any color image in other formats.</br> b. Find RED, GREEN and BLUE plane of the color image.</br> c. Convert color image into gray scale image and binary image.</br> d. Resize the image by one half and one quarter.</br> e. Image rotates by 45, 90 and 180 degrees.</br>

We are using the conventional image of Lena popularaly used in this field

In [1]:

```
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
import numpy as np
import cv2
```

In [2]:

```
lena = mpimg.imread("lena.png")
```

In [3]:

```
lena.shape
```

Out[3]:

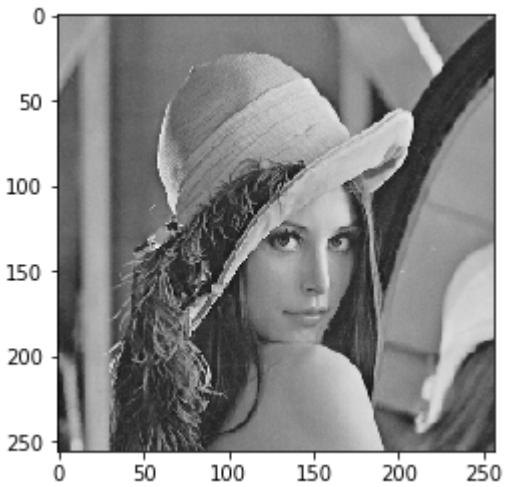
```
(256, 256, 3)
```

In [4]:

```
plt.imshow(lena)
```

Out[4]:

```
<matplotlib.image.AxesImage at 0x1217b3290>
```



In [5]:

```
sum(lena[0][0])/3
```

Out[5]:

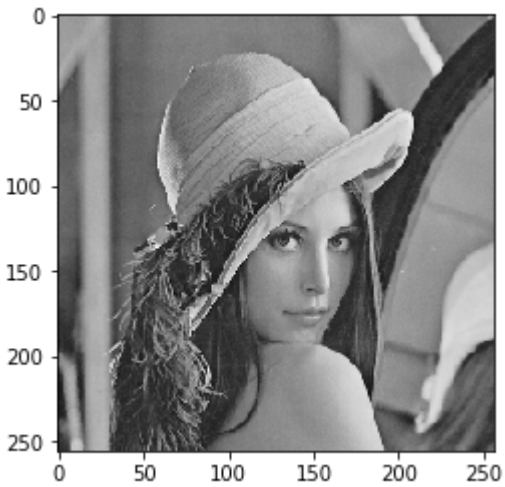
```
0.6313725709915161
```

In [6]:

```
plt.imshow(lena)
```

Out[6]:

```
<matplotlib.image.AxesImage at 0x121caf7d0>
```



In [ ]:

```

```

In [7]:

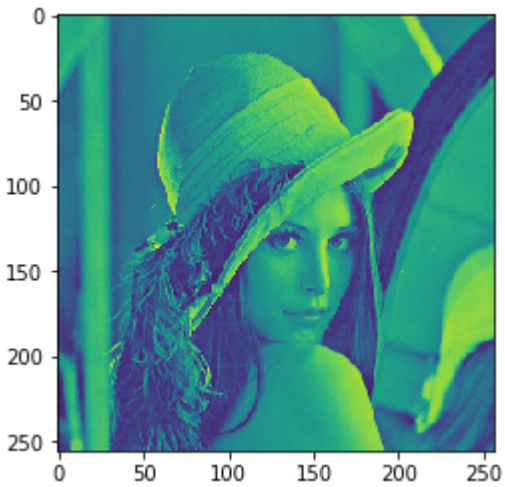
```
temp1 =[]
temp2 = []
for i in range(256):
    for j in range(256):
        temp2.append(sum(lena[i][j])/3)
    temp1.append(temp2)
    temp2 =[]
```

In [8]:

```
plt.imshow(temp1)
```

Out[8]:

```
<matplotlib.image.AxesImage at 0x1220a3550>
```



In [9]:

```
lena.shape
```

Out[9]:

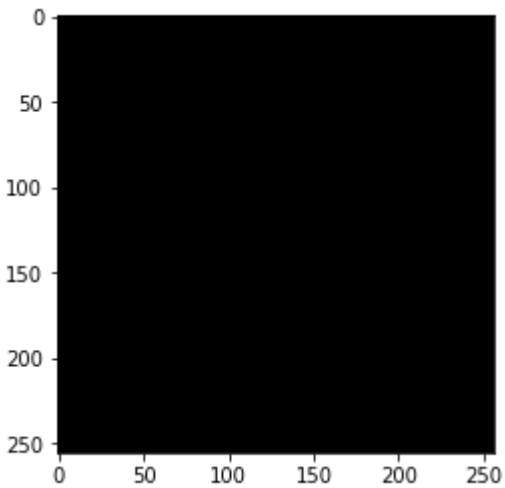
```
(256, 256, 3)
```

In [10]:

```
plt.imshow(temp1, cmap='gray', vmin=0, vmax=255)
```

Out[10]:

```
<matplotlib.image.AxesImage at 0x1221f5550>
```

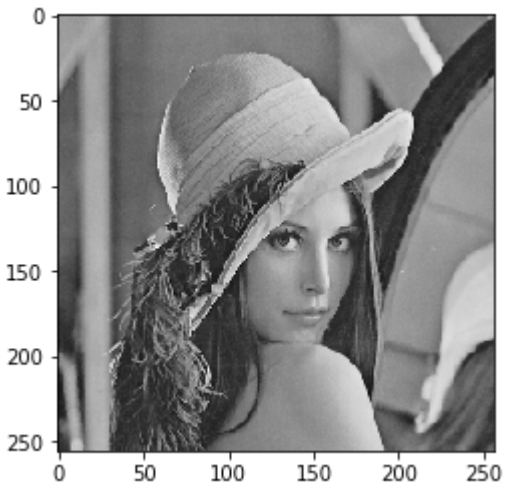


In [30]:

```
plt.imshow(lena, cmap='gray', vmin=0, vmax=255)
```

Out[30]:

```
<matplotlib.image.AxesImage at 0x129132a10>
```



In [11]:

```
a = np.array(temp1)
```

In [12]:

```
a.shape
```

Out[12]:

```
(256, 256)
```

In [13]:

```
a[0][0]
```

Out[13]:

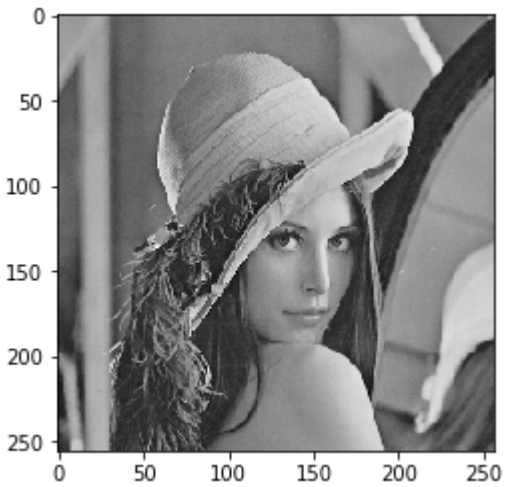
```
0.6313725709915161
```

In [14]:

```
plt.imshow(a, cmap='gray', vmin=0, vmax=1)
```

Out[14]:

```
<matplotlib.image.AxesImage at 0x121b47950>
```



In [15]:

```
a
```

Out[15]:

```
array([[0.63137257, 0.63137257, 0.6156863 , ..., 0.66274512, 0.66274
512,
        0.49411765],
       [0.63137257, 0.63137257, 0.6156863 , ..., 0.67058825, 0.66274
512,
        0.49411765],
       [0.63921571, 0.60784316, 0.62352943, ..., 0.58431375, 0.48627
451,
        0.24313726],
       ...,
       [0.21176471, 0.18039216, 0.20392157, ..., 0.34509805, 0.36078
432,
        0.35294119],
       [0.16470589, 0.18039216, 0.18039216, ..., 0.3764706 , 0.40784
314,
        0.38431373],
       [0.17254902, 0.19607843, 0.18039216, ..., 0.40784314, 0.40784
314,
        0.42352942]])
```

In [16]:

```
b = a*256
```

In [17]:

```
b
```

Out[17]:

```
array([[161.63137817, 161.63137817, 157.61569214, ..., 169.66275024,
        169.66275024, 126.49411774],
       [161.63137817, 161.63137817, 157.61569214, ..., 171.67059326,
        169.66275024, 126.49411774],
       [163.63922119, 155.60784912, 159.62353516, ..., 149.58432007,
        124.48627472,  62.24313736],
       ...,
       [ 54.21176529,  46.18039322,  52.20392227, ...,  88.3451004 ,
         92.36078644,  90.35294342],
       [ 42.16470718,  46.18039322,  46.18039322, ...,  96.37647247,
        104.40784454,  98.38431549],
       [ 44.1725502 ,  50.19607925,  46.18039322, ..., 104.40784454,
        104.40784454, 108.42353058]])
```

In [18]:

```
lena_Greyscale = b
```

In [19]:

```
lena_Greyscale.shape
```

Out[19]:

```
(256, 256)
```

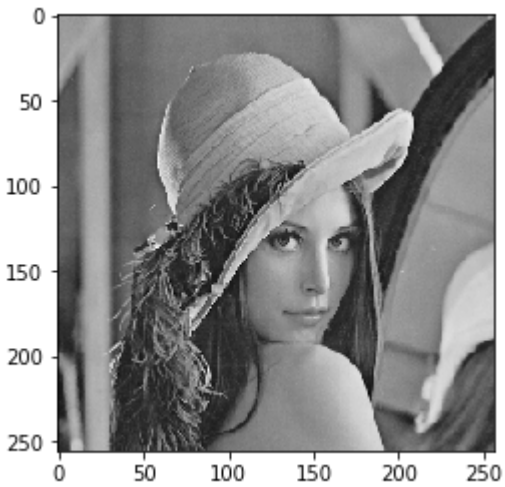
This is the converted Grayscale Image acheved by array maupulation

In [22]:

```
plt.imshow(a, cmap='gray')
```

Out[22]:

```
<matplotlib.image.AxesImage at 0x123e20d10>
```



In [27]:

```
lena.shape[1]
```

Out[27]:

```
256
```

This is the re-sized image of 125 Px.

In [28]:

```
resized = cv2.resize(lena_Greyscale, (lena_Greyscale.shape[0] * 50//100,lena_Gre
yscale.shape[1] * 50//100))
```

In [29]:

```
plt.imshow(resized, cmap='gray')
```

Out[29]:

```
<matplotlib.image.AxesImage at 0x1242bb550>
```

Rotation of the image by 45 90 and 180 degree

In [31]:

```
center = (lena_Greyscale.shape[0]//2 , lena_Greyscale.shape[1]//2)
```

In [38]:

```
mask = cv2.getRotationMatrix2D(center, 45, 1)
```

In [41]:

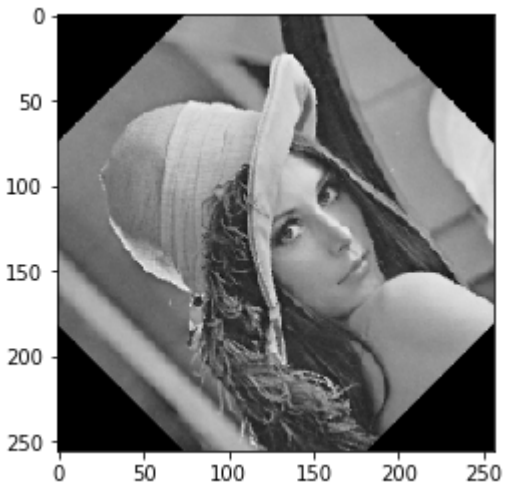
```
rotated45 = cv2.warpAffine(lena_Greyscale, mask, (lena_Greyscale.shape[0],lena_G
reyscale.shape[1]))
```

In [42]:

```
plt.imshow(rotated45, cmap='gray')
```

Out[42]:

```
<matplotlib.image.AxesImage at 0x123662750>
```



In [47]:

```
mask90 = cv2.getRotationMatrix2D(center, 90, 1)
```

In [44]:

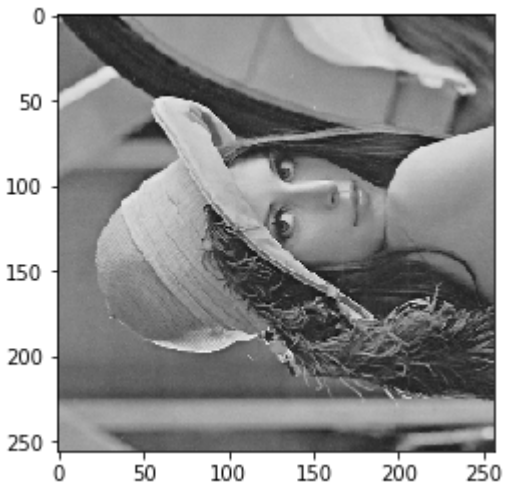
```
rotated90 = cv2.warpAffine(lena_Greyscale, mask90, (lena_Greyscale.shape[0],lena
_Greyscale.shape[1]))
```

In [45]:

```
plt.imshow(rotated90, cmap='gray')
```

Out[45]:

```
<matplotlib.image.AxesImage at 0x12370fb50>
```



In [48]:

```
mask180 = cv2.getRotationMatrix2D(center, 180, 1)
```

In [49]:

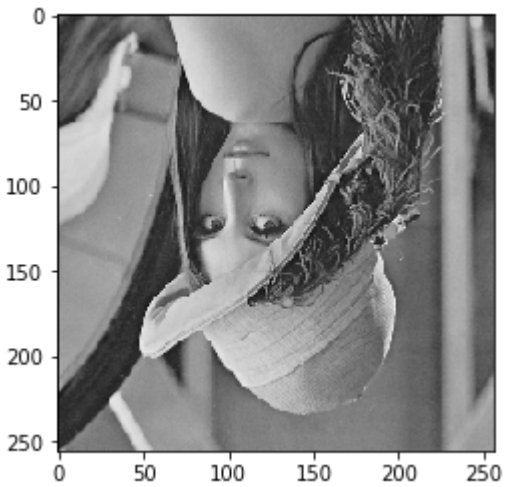
```
rotated180 = cv2.warpAffine(lena_Greyscale, mask180, (lena_Greyscale.shape[0],le
na_Greyscale.shape[1]))
```

In [50]:

```
plt.imshow(rotated180, cmap='gray')
```

Out[50]:

```
<matplotlib.image.AxesImage at 0x12405cbd0>
```



Question 2

Create black and white images (A) of size 1024x1024. Which consists of alternativ horizontal lines of black and white? Each line is of size 128.
Create black and white images (B) of size 1024x1024. Which consists of alternative vertical lines of black and white? Each line is of size128.Perform the following operations on Image A and Image B.
a. Image addition of A and B
b. Subtraction of A and B
c. Multiplying Images of A and B
a. Create a grayscale image of size 256x1024. Intensity of image should vary sinusoidal.
b. Create a white image of size 256x256, with black box of size 58x58 at centre.

In [81]:

```
Image_A_Array = []
for i in range(1024):
    temps = []
    for j in range(1024):
        temps.append(1)
    Image_A_Array.append(temps)
```

In [82]:

```
Image_A_Array = np.array(Image_A_Array)
```

In [83]:

```
Image_A_Array.shape
```

Out[83]:

```
(1024, 1024)
```

In [84]:

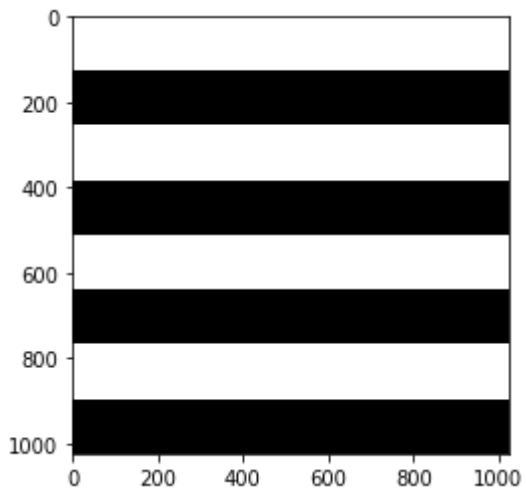
```
temp_index = 0
for i in range (1, (1024//128)+1):
    if i % 2 ==0:
        for j in range (128):
            Image_A_Array[temp_index] = Image_A_Array[temp_index]*0
            temp_index+=1
    if i % 2 ==1:
        for j in range (128):
            Image_A_Array[temp_index] = Image_A_Array[temp_index]*255
            temp_index+=1
```

In [85]:

```
plt.imshow(Image_A_Array, cmap='gray')
```

Out[85]:

```
<matplotlib.image.AxesImage at 0x127b44290>
```



In [87]:

```
Image_A_Array.max()
```

Out[87]:

```
255
```

In [88]:

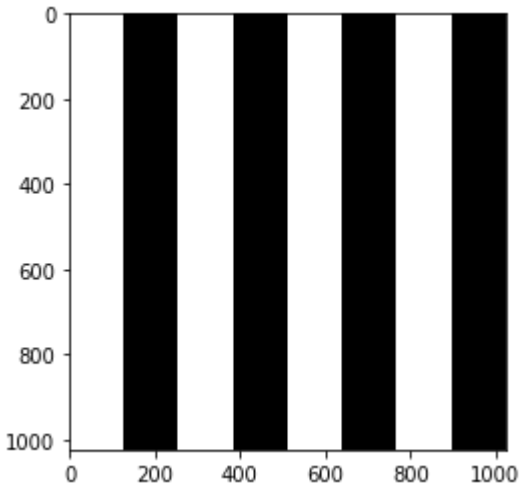
```
Image_B_Array = Image_A_Array.transpose()
```

In [89]:

```
plt.imshow(Image_B_Array, cmap='gray')
```

Out[89]:

```
<matplotlib.image.AxesImage at 0x123145ed0>
```



In [90]:

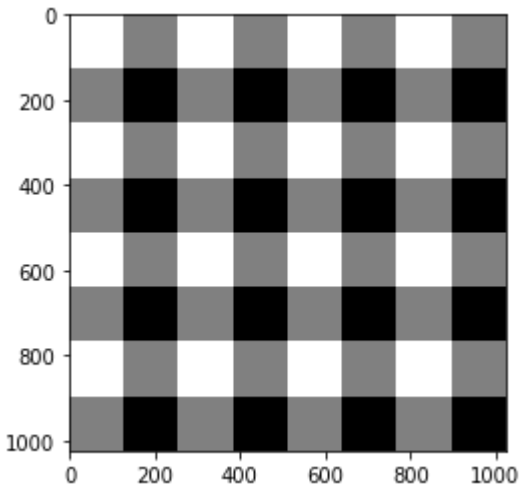
```
Added_Array = Image_A_Array + Image_B_Array
```

In [91]:

```
plt.imshow(Added_Array, cmap='gray')
```

Out[91]:

```
<matplotlib.image.AxesImage at 0x12a1ccbd0>
```



In [92]:

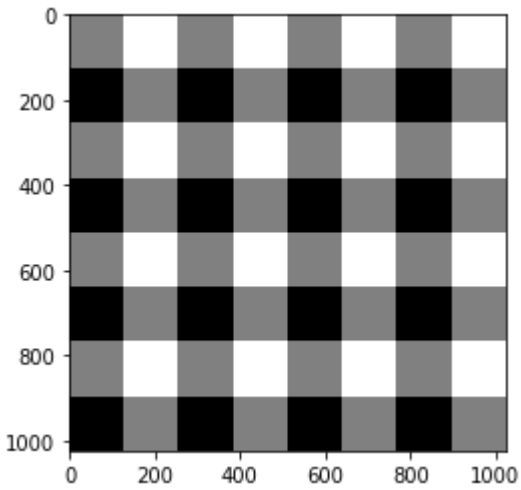
```
Substracted_Array = Image_A_Array - Image_B_Array
```

In [93]:

```
plt.imshow(Substracted_Array, cmap='gray')
```

Out[93]:

```
<matplotlib.image.AxesImage at 0x12a1da110>
```



In [94]:

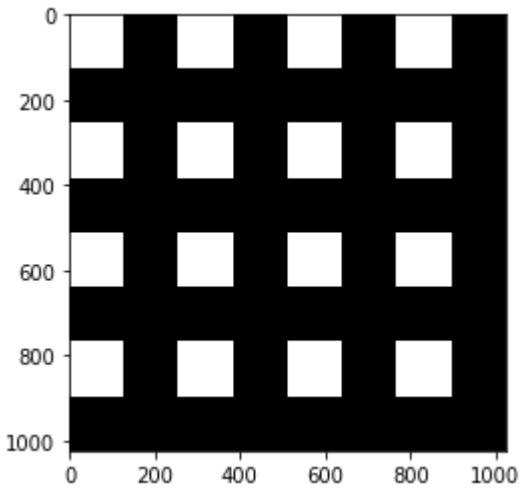
```
multiplied_array = Image_A_Array*Image_B_Array
```

In [95]:

```
plt.imshow(multiplied_array, cmap='gray')
```

Out[95]:

```
<matplotlib.image.AxesImage at 0x1259b4450>
```



In [97]:

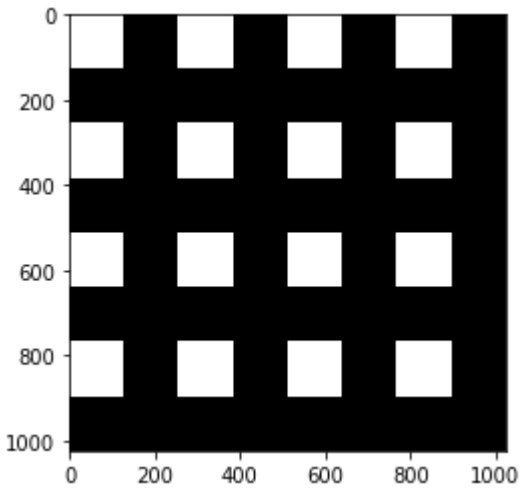
```
arry_multiply_array = Image_A_Array.dot(Image_B_Array)
```

In [98]:

```
plt.imshow(arry_multiply_array, cmap='gray')
```

Out[98]:

```
<matplotlib.image.AxesImage at 0x12c45fb10>
```



In [105]:

```
sine_image_Array = []
for i in range(256):
    temp= []
    for j in range(1024):
        temp.append(np.sin(np.deg2rad(j)))
    sine_image_Array.append(temp)
```

In [106]:

```
sine_image_Array = np.array(sine_image_Array)
```

In [107]:

```
sine_image_Array.shape
```

Out[107]:

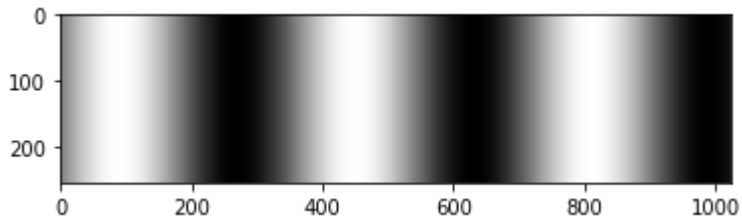
```
(256, 1024)
```

In [108]:

```
plt.imshow(sine_image_Array, cmap='gray')
```

Out[108]:

```
<matplotlib.image.AxesImage at 0x12ec02850>
```



In [115]:

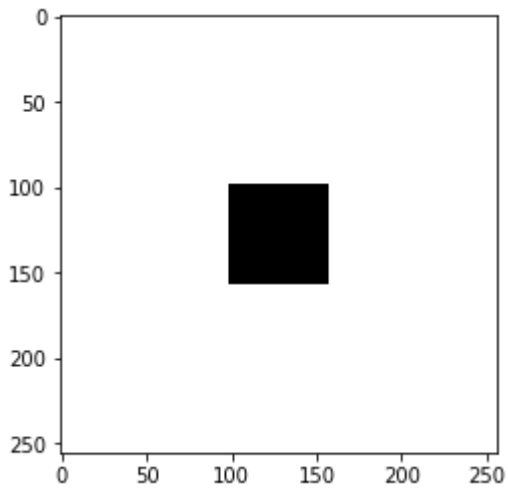
```
box_image = []
for i in range(256):
    temp =[]
    for j in range(256):
        if i>=99 and i<157 and j>=99 and j<157:
            temp.append(0)
        else:
            temp.append(255)
    box_image.append(temp)
```

In [116]:

```
plt.imshow(box_image, cmap='gray')
```

Out[116]:

```
<matplotlib.image.AxesImage at 0x12566ec90>
```



Question 3

Develop programs for following intensity transformation operation on a gray scale image. Collect any gray scale image from any source. Process that image using these operations.
a. Image negative
b. Log transformation and inverse log transform: s = c log (1+r), c is a const, r ≥ 0. s is pixel intensity of output image, r is the pixel intensity of input image. Study thev effect of constant c on the quality of output image.
c. Power law transformation: Study the effect of different values of Gamma used in this transformation.
d. Contrast stretching
e. Gray level slicing

Image Negative

In [122]:

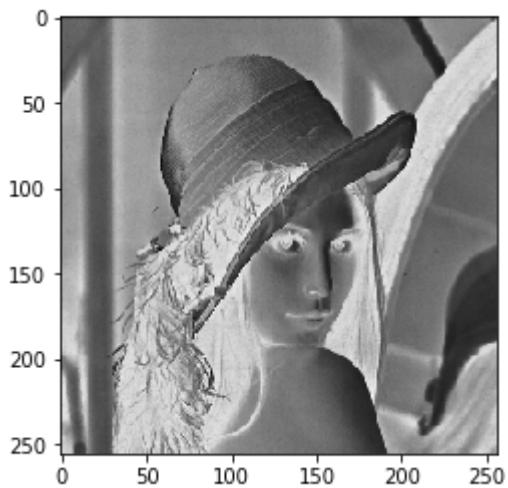
```
negative_image = []
for i in lena_Greyscale:
    temp = []
    for j in i:
        temp.append(255 - j)
    negative_image.append(temp)
```

In [123]:

```
plt.imshow(negative_image, cmap='gray')
```

Out[123]:

```
<matplotlib.image.AxesImage at 0x12f41bdd0>
```



Log Transform

In [147]:

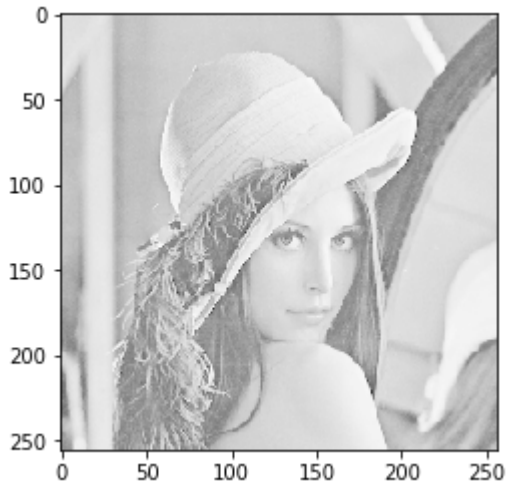
```
log_image = []
for i in lena_Greyscale:
    temp = []
    for j in i:
        temp.append(0.01*np.log(1+j))
    log_image.append(temp)
```

In [148]:

```
plt.imshow(log_image, cmap='gray')
```

Out[148]:

```
<matplotlib.image.AxesImage at 0x1311d7c10>
```



In [149]:

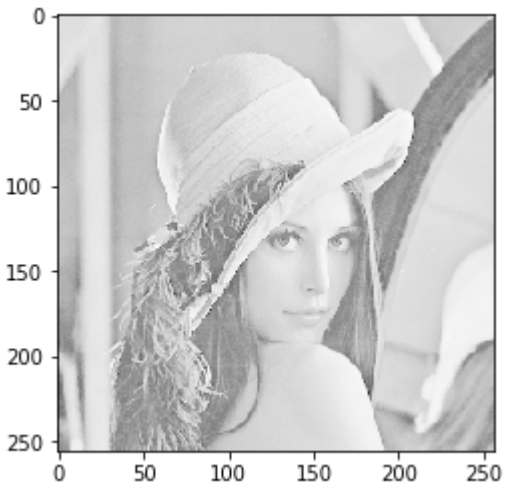
```
log_image = []
for i in lena_Greyscale:
    temp = []
    for j in i:
        temp.append(1000*np.log(1+j))
    log_image.append(temp)
```

In [150]:

```
plt.imshow(log_image, cmap='gray')
```

Out[150]:

```
<matplotlib.image.AxesImage at 0x1304cec50>
```



Power Law with different gamma Values

In [151]:

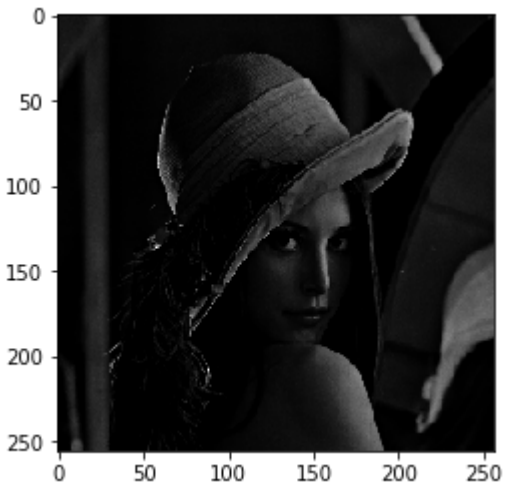
```
power_law_image = []
for i in lena_Greyscale:
    temp = []
    for j in i:
        temp.append(1*j**5)
    power_law_image.append(temp)
```

In [152]:

```
plt.imshow(power_law_image, cmap='gray')
```

Out[152]:

```
<matplotlib.image.AxesImage at 0x1313d1510>
```



In [154]:

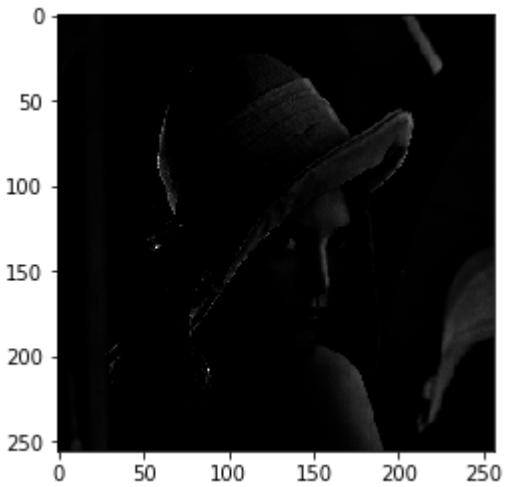
```
power_law_image = []
for i in lena_Greyscale:
    temp = []
    for j in i:
        temp.append(1*j**10)
    power_law_image.append(temp)
```

In [155]:

```
plt.imshow(power_law_image, cmap='gray')
```

Out[155]:

```
<matplotlib.image.AxesImage at 0x12508f890>
```



In [159]:

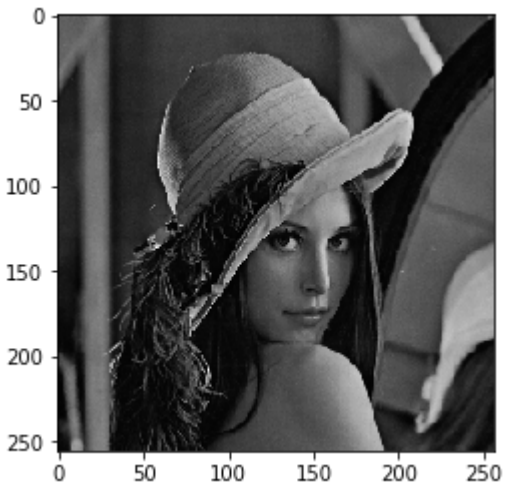
```
power_law_image = []
for i in lena_Greyscale:
    temp = []
    for j in i:
        temp.append(1*j**2)
    power_law_image.append(temp)
```

In [160]:

```
plt.imshow(power_law_image, cmap='gray')
```

Out[160]:

```
<matplotlib.image.AxesImage at 0x1312fa550>
```



Contrast Streaching

In [163]:

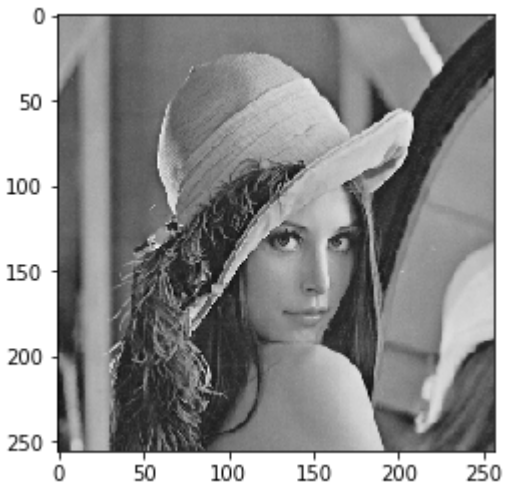
```
contast_streaching_image = []
for i in lena_Greyscale:
    temp = []
    for j in i:
        temp.append(j-1000)
    contast_streaching_image.append(temp)
```

In [164]:

```
plt.imshow(contast_streaching_image, cmap='gray')
```

Out[164]:

```
<matplotlib.image.AxesImage at 0x128e51610>
```



In [165]:

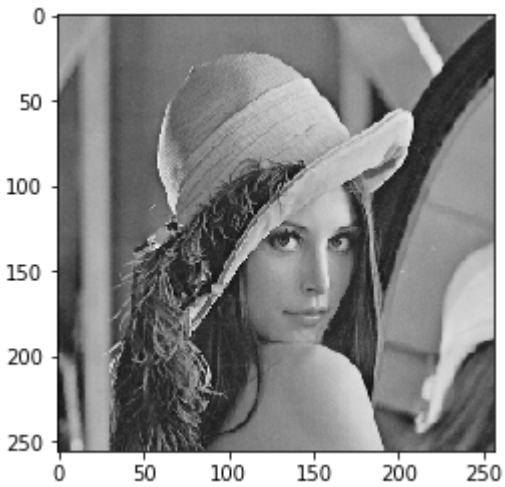
```
contast_streaching_image = []
for i in lena_Greyscale:
    temp = []
    for j in i:
        temp.append(j-6)
    contast_streaching_image.append(temp)
```

In [166]:

```
plt.imshow(contast_streaching_image, cmap='gray')
```

Out[166]:

```
<matplotlib.image.AxesImage at 0x131a299d0>
```



Grey Level Slicing

In [167]:

```
Grey_level_image = []
for i in lena_Greyscale:
    temp = []
    for j in i:
        if j<2:
            temp.append(0)
        else:
            temp.append(j)
    Grey_level_image.append(temp)
```

In [ ]:

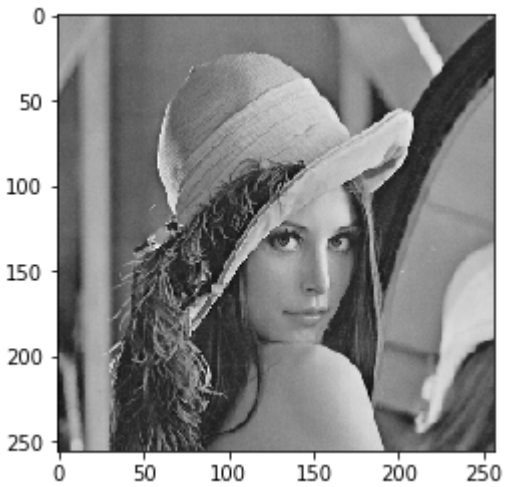
```

```

In [168]:

```
plt.imshow(Grey_level_image, cmap='gray')
```

Out[168]:

```
<matplotlib.image.AxesImage at 0x131a39250>
```



In [ ]: