DIP Lab Assignment 02

Question 1

a. Read, Display and write any color image in other formats.

In [2]:

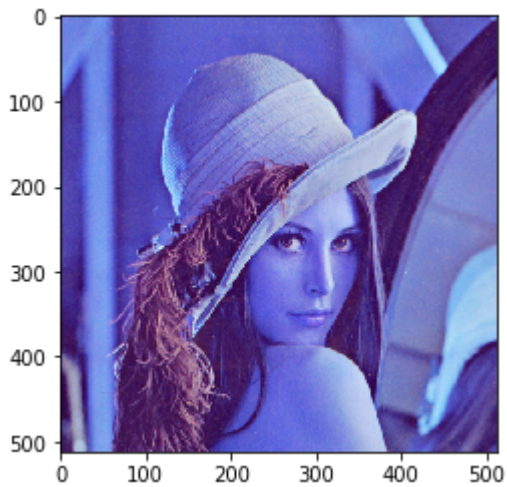
```
import matplotlib.pyplot as plt
import numpy as np
import cv2
import os
```

In [3]:

```
img = cv2.imread("lena_color.tiff")
```

In [4]:

```
plt.imshow(img, cmap=plt.cm.bone)
plt.show()
```



In [5]:

```
img.shape
```

Out[5]:

```
(512, 512, 3)
```

In [6]:

```
imgg = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

In [7]:

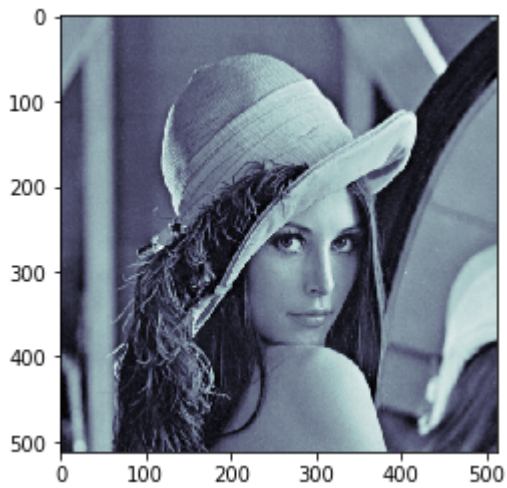
```
imgg.shape
```

Out[7]:

```
(512, 512)
```

In [8]:

```
plt.imshow(imgg, cmap=plt.cm.bone)
plt.show()
```



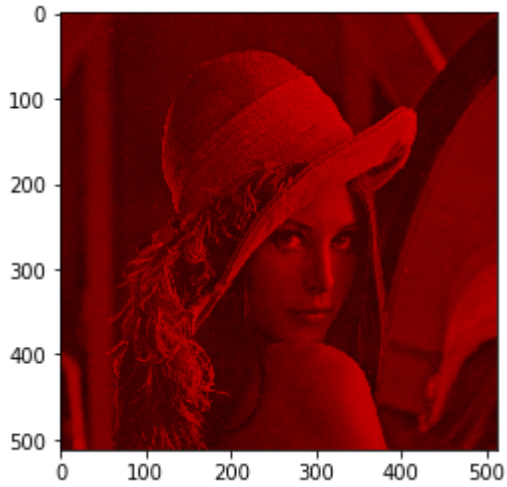
b. Find RED, GREEN and BLUE plane of the color image.

In [9]:

```
red = np.zeros((img.shape),np.uint8)
red[:,:,0] = img[:,:,0]
```

In [10]:

```
plt.imshow(red, cmap=plt.cm.bone)
plt.show()
```

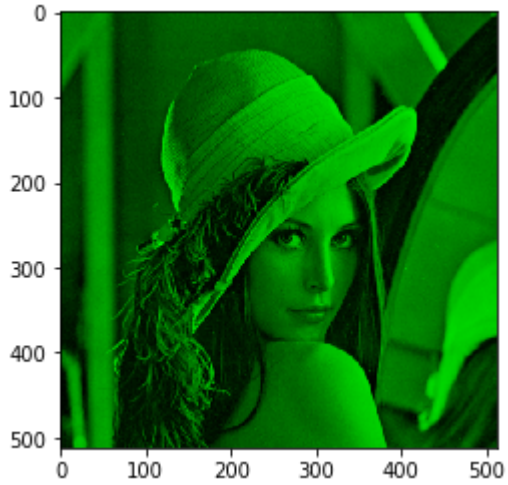


In [11]:

```
green = np.zeros((img.shape),np.uint8)
green[:,:,1] = img[:,:,1]
```

In [12]:

```
plt.imshow(green, cmap=plt.cm.bone)
plt.show()
```

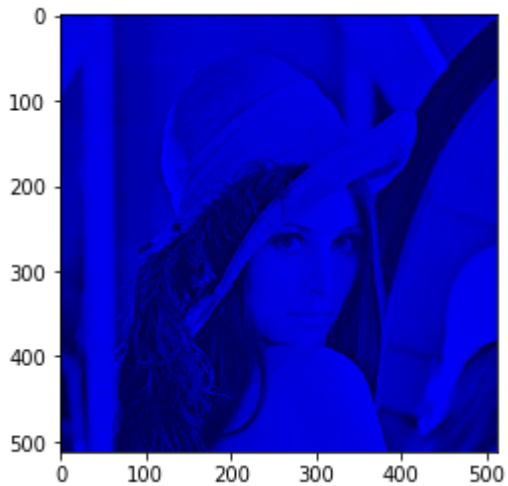


In [13]:

```
blue = np.zeros((img.shape),np.uint8)
blue[:,:,2] = img[:,:,2]
```

In [14]:

```
plt.imshow(blue, cmap=plt.cm.bone)
plt.show()
```



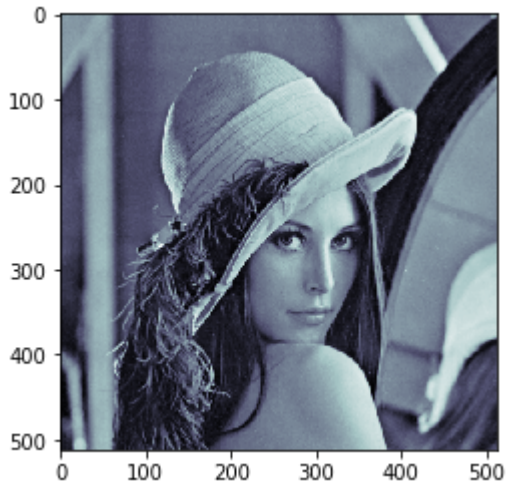
c. Convert color image into gray scale image and binary image.

In [15]:

```
imgg = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

In [16]:

```
plt.imshow(imgg, cmap=plt.cm.bone)
plt.show()
```

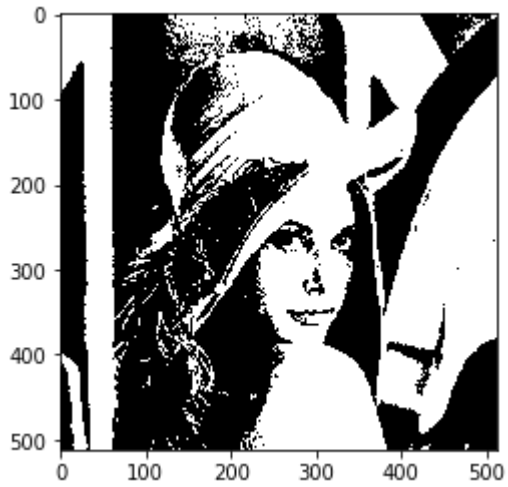


In [17]:

```
thresh, imgb = cv2.threshold(imgg, 127, 255, cv2.THRESH_BINARY)
```

In [18]:

```
plt.imshow(imgb, cmap=plt.cm.bone)
plt.show()
```



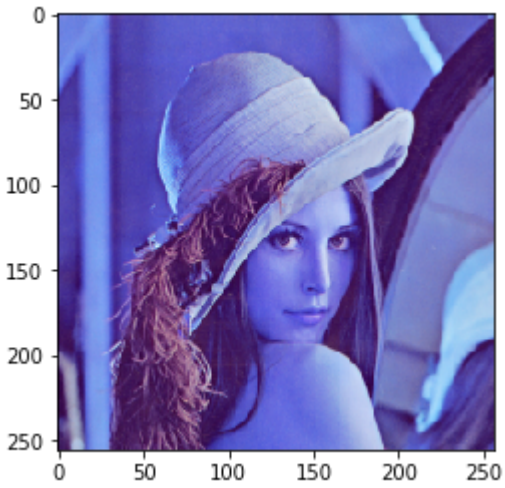
d. Resize the image by one half and one quarter.

In [19]:

```
imghalf = cv2.resize(img, (256,256), interpolation = cv2.INTER_AREA)
```

In [20]:

```
plt.imshow(imghalf, cmap=plt.cm.bone)
plt.show()
```

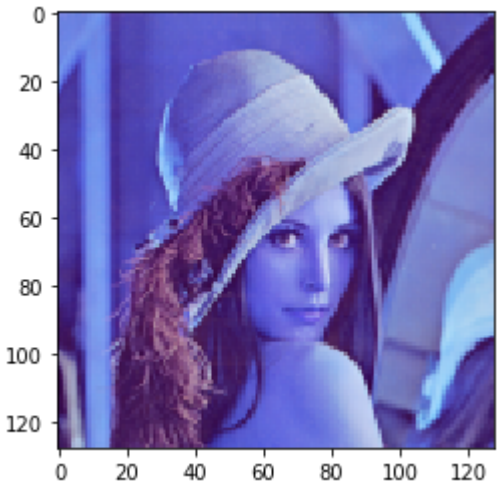


In [21]:

```
imgquarter = cv2.resize(img, (128,128), interpolation = cv2.INTER_AREA)
```

In [22]:

```
plt.imshow(imgquarter, cmap=plt.cm.bone)
plt.show()
```



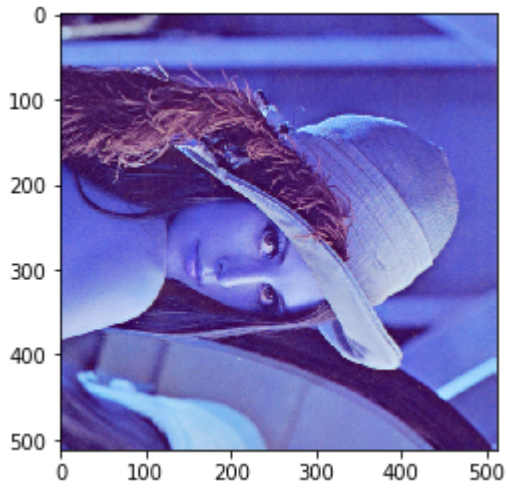
e. Image rotates by 45, 90 and 180 degrees.

In [23]:

```
img90 = cv2.rotate(img, cv2.ROTATE_90_CLOCKWISE)
```

In [24]:

```
plt.imshow(img90, cmap=plt.cm.bone)
plt.show()
```

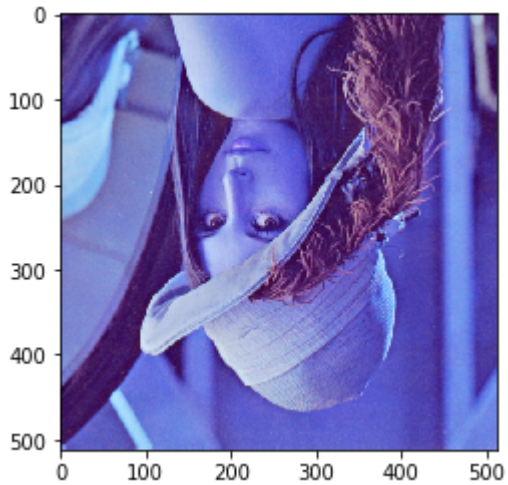


In [25]:

```
img180 = cv2.rotate(img, cv2.ROTATE_180)
```

In [26]:

```
plt.imshow(img180, cmap=plt.cm.bone)
plt.show()
```

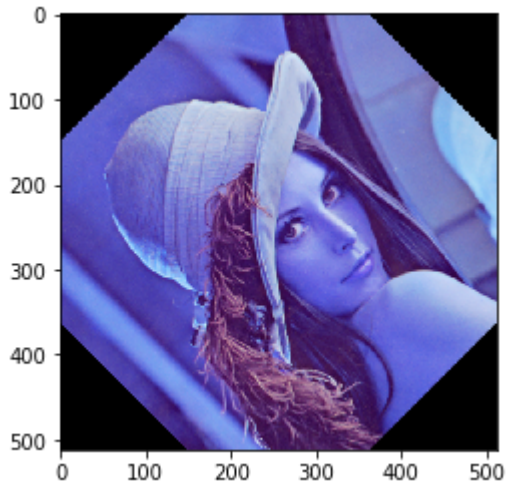


In [27]:

```
rows,cols,panel = img.shape
inm = cv2.getRotationMatrix2D((cols/2,rows/2),45,1)
img45 = cv2.warpAffine(img,inm,(cols,rows))
```

In [28]:

```
plt.imshow(img45, cmap=plt.cm.bone)
plt.show()
```



Question 2

In [29]:

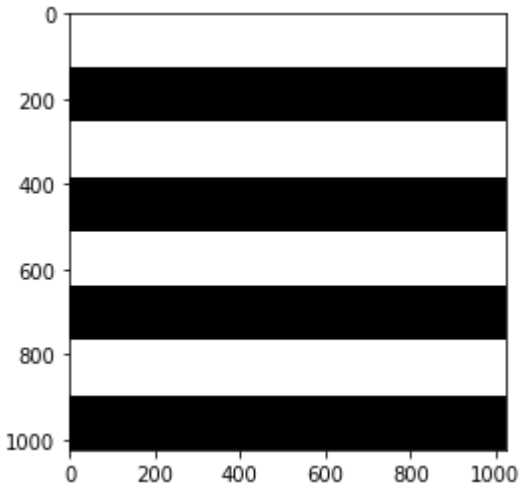
```
A = np.zeros((1024,1024))
```

In [30]:

```
c = 1
i,j = 0,0
while(i<1024):
    for j in range(1024):
        A[i][j] = 1
    i += 1
    if(i==c*128):
        i += 128
        c += 2
```

In [31]:

```
plt.imshow(A, cmap=plt.cm.bone)
plt.show()
```



In [32]:

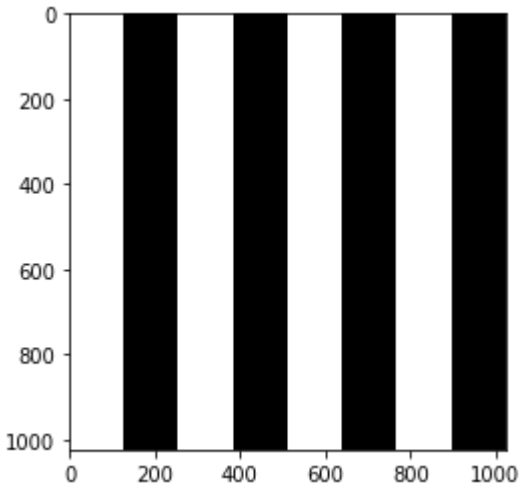
```
B = np.zeros((1024,1024))
```

In [33]:

```
c = 1
i,j = 0,0
while(i<1024):
    for j in range(1024):
        B[j][i] = 1
    i += 1
    if(i==c*128):
        i += 128
        c += 2
```

In [34]:

```
plt.imshow(B, cmap=plt.cm.bone)
plt.show()
```



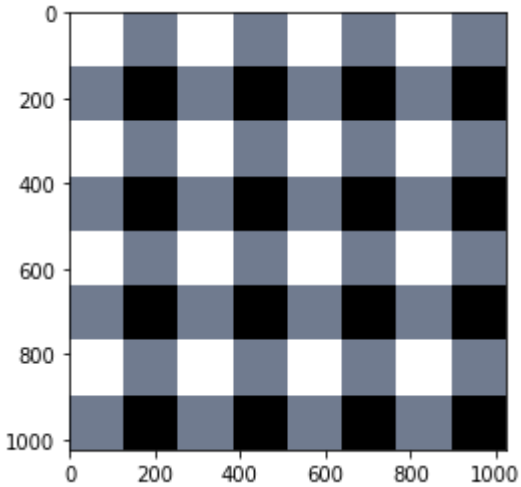
a. Image addition of A and B

In [35]:

```
add = np.add(A,B)
```

In [36]:

```
plt.imshow(add, cmap=plt.cm.bone)
plt.show()
```



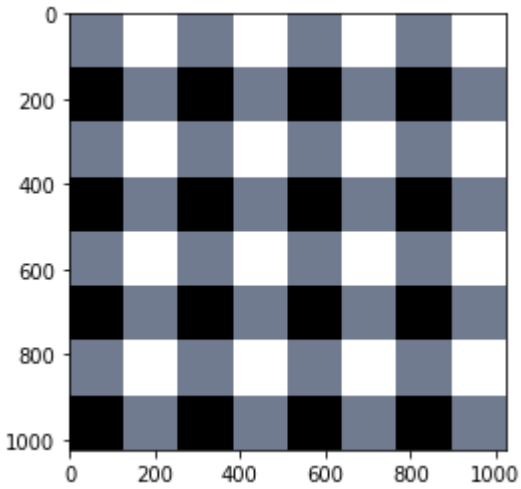
b. Subtraction of A and B

In [37]:

```
sub = np.subtract(A,B)
```

In [38]:

```
plt.imshow(sub, cmap=plt.cm.bone)
plt.show()
```



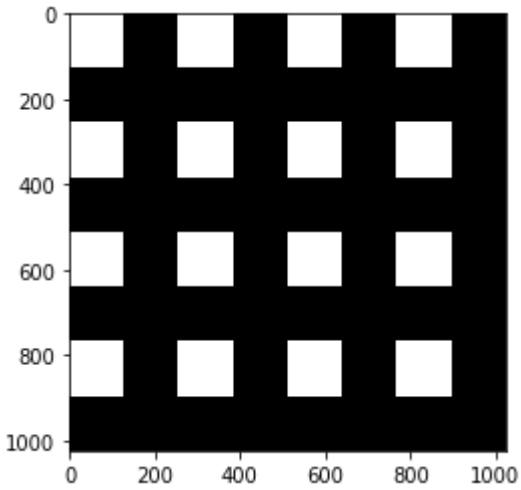
c. Multiplying Images of A and B

In [39]:

```
mul = np.multiply(A,B)
```

In [40]:

```
plt.imshow(mul, cmap=plt.cm.bone)
plt.show()
```



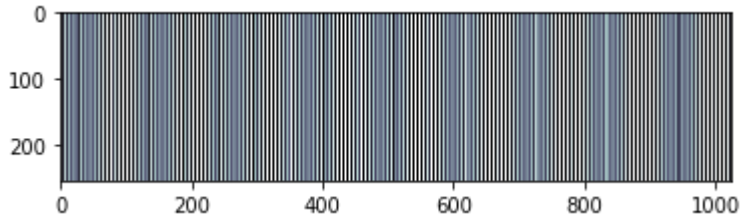
d. Create a grayscale image of size 256x1024. Intensity of image should vary sinusoidal.

In [41]:

```
sin = np.zeros((256,1024))
for i in range(256):
    for j in range(1024):
        sin[i][j] = np.sin(j)
```

In [42]:

```
plt.imshow(sin, cmap=plt.cm.bone)
plt.show()
```



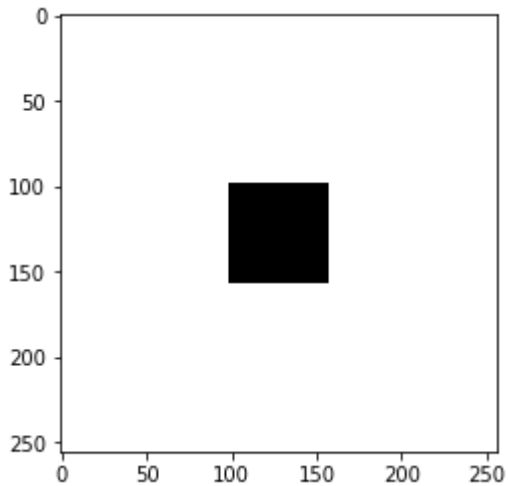
e. Create a white image of size 256x256, with black box of size 58x58 at centre.

In [43]:

```
wibb = np.ones((256,256))
pad = (256-58)//2
for i in range(pad,(256-pad)):
    for j in range(pad,(256-pad)):
        wibb[i][j] = 0
```

In [44]:

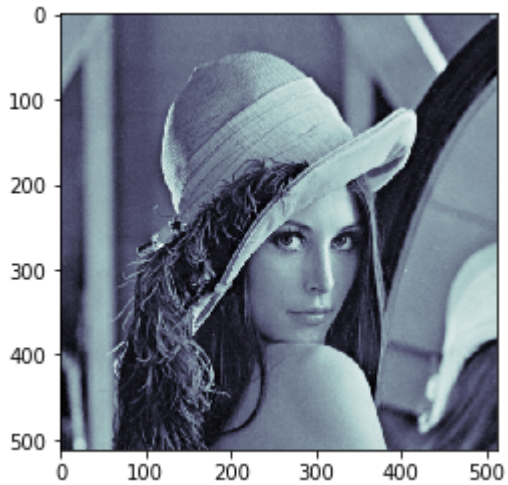
```
plt.imshow(wibb, cmap=plt.cm.bone)
plt.show()
```



Question 3

In [45]:

```
plt.imshow(imgg, cmap=plt.cm.bone)
plt.show()
```



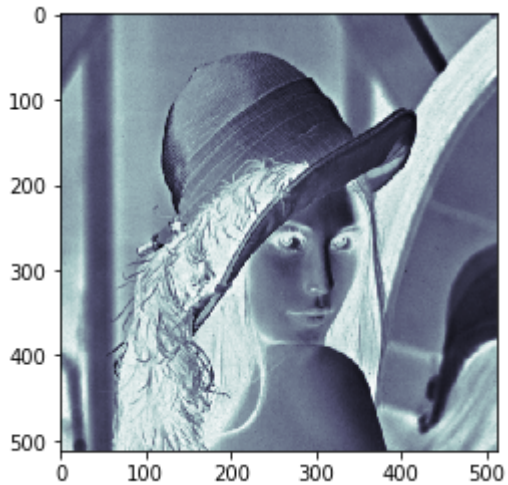
a. Image negative

In [46]:

```
imggn = np.zeros((512,512))
for i in range(512):
    for j in range(512):
        imggn[i][j] = (255 - imgg[i][j])
```

In [47]:

```
plt.imshow(imggn, cmap=plt.cm.bone)
plt.show()
```



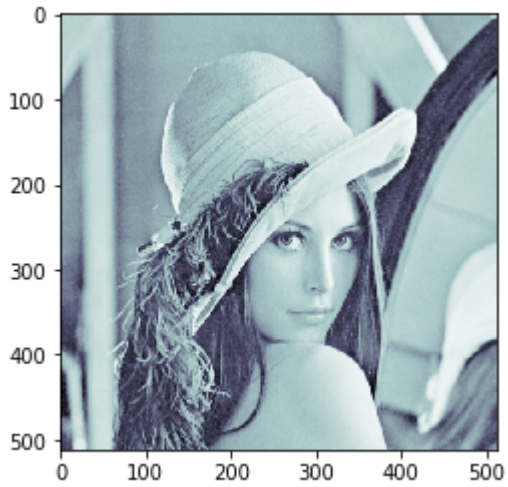
b. Log transformation and inverse log transform: s = c log (1+r), c is a const, r ≥ 0. s is pixel intensity of output image, r is the pixel intensity of input image. Study the effect of constant c on the quality of output image.

In [48]:

```
imggl = np.zeros((512,512))
c = 1
for i in range(512):
    for j in range(512):
        imggl[i][j] = c*np.log(1+imgg[i][j])
```

In [49]:

```
plt.imshow(imggl, cmap=plt.cm.bone)
plt.show()
```

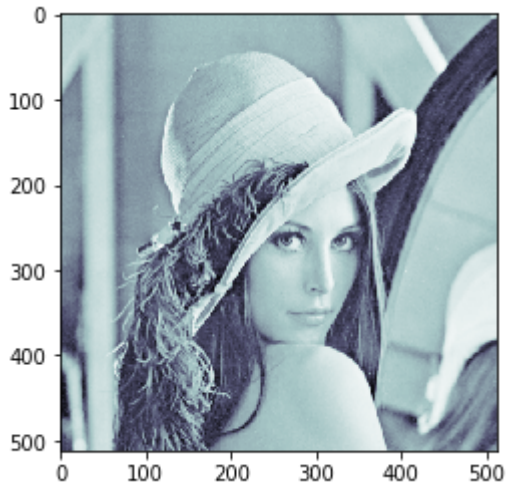


In [52]:

```
imggil = np.zeros((512,512))
c = 1
for i in range(512):
    for j in range(512):
        imggil[i][j] = (imgg[i][j])**(np.log(256) / (255)) - 1
```

In [54]:

```
plt.imshow(imggil, cmap=plt.cm.bone)
plt.show()
```

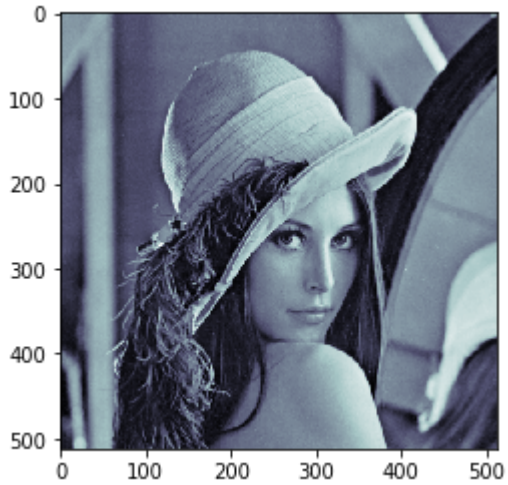


c. Power law transformation: Study the effect of different values of Gamma used in this transformation.

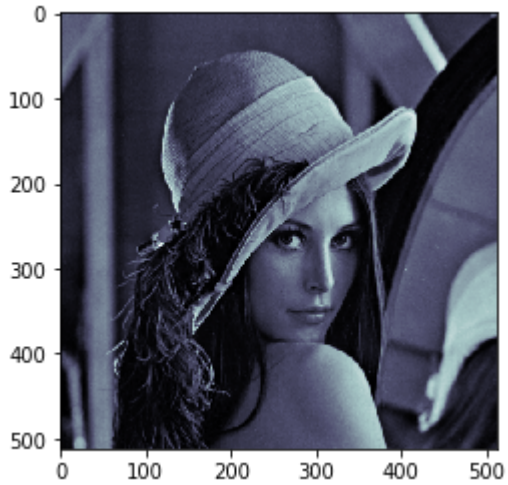
In [107]:

```
imggp = np.zeros((512,512))
c = 1
for k in range(1,11):
    for i in range(512):
        for j in range(512):
            imggp[i][j] = c*(imgg[i][j])**k
    print("\nFor Gamma: ",k)
    plt.imshow(imggp, cmap=plt.cm.bone)
    plt.show()
```

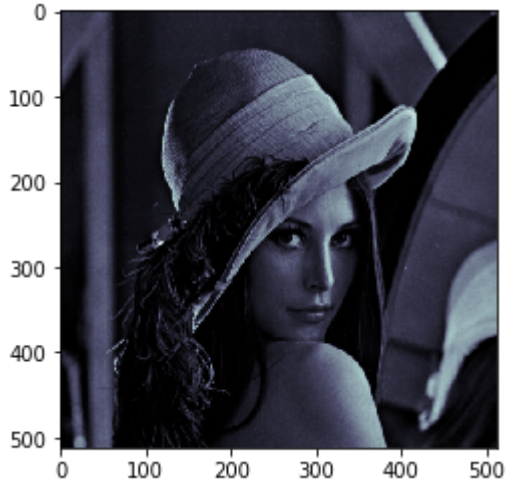
```
For Gamma:  1
```



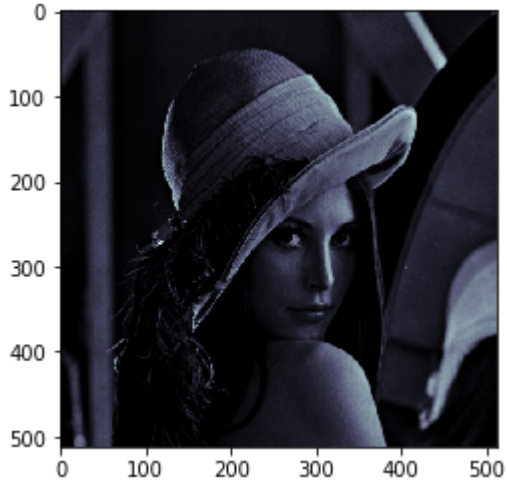

```
For Gamma:  2
```



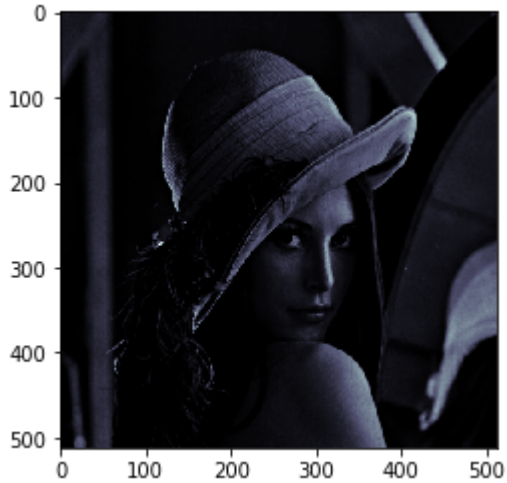

```
For Gamma:  3
```



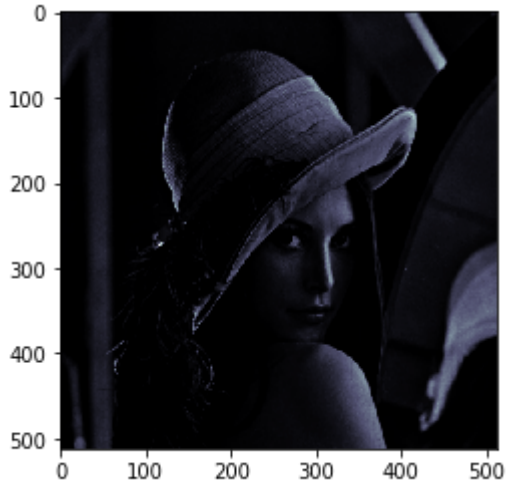

```
For Gamma:  4
```



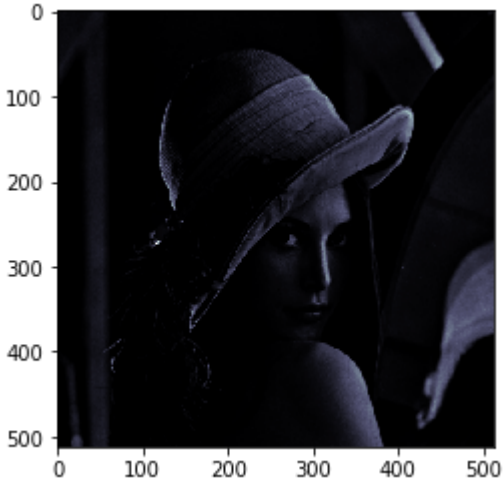

```
For Gamma:  5
```



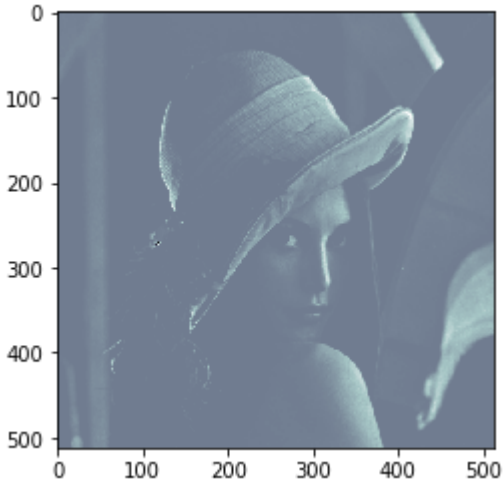

```
For Gamma:  6
```



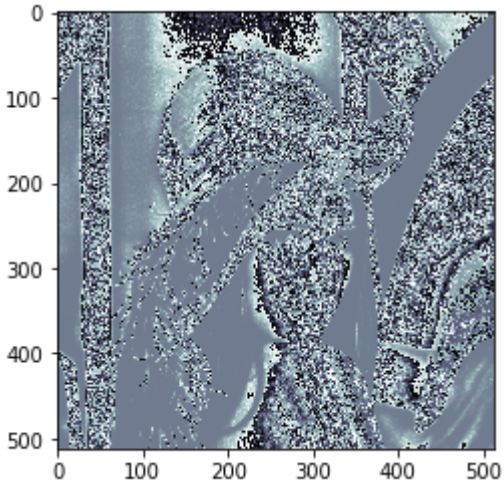

```
For Gamma:  7
```



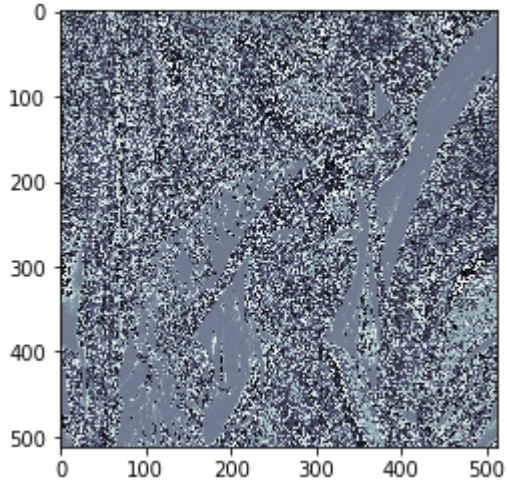

```
For Gamma:  8
```




```
For Gamma:  9
```




```
For Gamma:  10
```



d. Contrast stretching

In [109]:

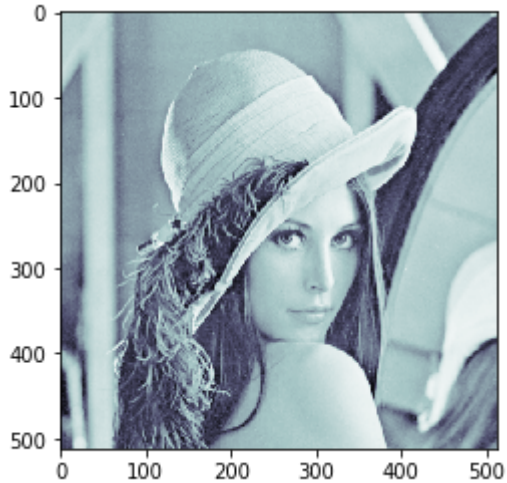
```
a = 0
b = 255
c = np.amin(imgg)
d = np.amax(imgg)
```

In [110]:

```
imggcs = np.zeros((512,512))
c = 1
for i in range(512):
    for j in range(512):
        imggcs[i][j] = (imgg[i][j]-c)*((b-a)/(d-c))+a
```

In [111]:

```
plt.imshow(imggcs, cmap=plt.cm.bone)
plt.show()
```



e. Gray level slicing

In [120]:

```
min_range = 10
max_range = 60
```

In [121]:

```
imgggls = np.zeros((512,512))
for i in range(512):
    for j in range(512):
        if imgg[i,j]>min_range and imgg[i,j]<max_range:
            imgggls[i,j] = 255
        else:
            imgggls[i,j] = 0
```

In [122]:

```
plt.imshow(imgggls, cmap=plt.cm.bone)
plt.show()
```