ques3

September 5, 2020

This code has been compiled by

Rohit Lal BT17ECE067 *https://rohitlal.live/*

1 Import Libraries

[1]:

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

2 Write Code to do Image operation

- An object oriented code that will do various required operations on image
- Code is optimised using `numpy` and `OpenCV` library
- Use Interactive Jupyter Notebook for code and image viewing

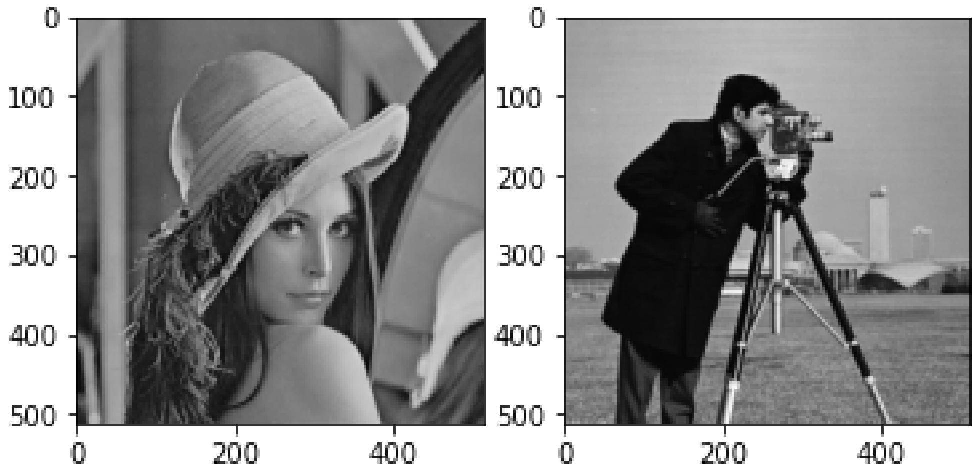
[2]:

```
class Assignment():

    def __init__(self, a,b):
        self.a = plt.imread(a)
        self.b = plt.imread(b)

    def show_images(self,a,b):
        plt.subplot(1,2,1)
        plt.imshow(a,cmap='gray', vmin=0, vmax=255)
        plt.subplot(1,2,2)
        plt.imshow(b,cmap='gray', vmin=0, vmax=255)
        plt.show()

    def transpose(self):
        # Question 3.a : Transpose of  images
        lenna_transpose = self.a.T
        camera_transpose = self.b.T
        return lenna_transpose,camera_transpose

    def inverse(self):
        # Question 3.b : Inverse of  images
        lenna_inv = np.linalg.inv(self.a)
        camera_inv = np.linalg.inv(self.b)
        return lenna_inv , camera_inv

    def add(self):
        # Question 3.c : Addition of images (both)
        add_1 = self.a + self.b
        add_2 = self.b + self.a
        return add_1,add_2

    def sub(self):
        # Question 3.d : Subtraction of images (both)
        sub_1 = self.a - self.b
        sub_2 = self.b - self.a
        return sub_1,sub_2

    def mul(self):
        # Question 3.e : Multiplication of images
        mul_1 = np.matmul(self.a, self.b)
        mul_2 = np.matmul(self.b, self.a)
        return mul_1,mul_2

    def mul_scalar(self,c):
        # Question 3.f 2.g: Multiplication of images by scalar greater and less than 1
        a_scalar = self.a.astype(int) * c
        a_scalar = np.clip(a_scalar,0,255).astype(np.uint8)

        b_scalar = self.b.astype(int) * c
        b_scalar = np.clip(b_scalar,0,255).astype(np.uint8)

        return a_scalar,b_scalar

    def elementwise_mul(self):
        # Question 2.h : Element by element multiplication
        elementwise_1 = np.multiply(a,b)
        elementwise_2 = np.multiply(b,a)
        return elementwise_1,elementwise_2

    def value_finder(self,key):
        # Question 2.i 2.j  : Find the specific value of X
        x1,y1 = np.where(self.a == key)
        print(f'In matrix a, found {key} at : [{x1[0]},{y1[0]}]' )

        x2,y2 = np.where(self.b == key)
        print(f'In matrix b, found {key} at : [{x2[0]},{y2[0]}]' )

    def find_and_replace(self, key, replace_by):
        # Question 2.k  : Find the specific value of X and replace it
        x1,y1 = np.where(self.a == key)
        a_copy = self.a.copy()
        b_copy = self.b.copy()
        for c,(i,j )in enumerate(zip(x1,y1)):
            a_copy[i][j] = replace_by
            print(f'In matrix a, replaced {key} at : [{i+1},{j+1}]' )
        print(f'In matrix a, {key} found {c} times' )

        x2,y2 = np.where(self.b == key)

        for c,(i,j) in enumerate(zip(x2,y2)):
            b_copy[i][j] = replace_by
            print(f'In matrix b, replaced {key} at : [{i+1},{j+1}]' )
        print(f'In matrix b, {key} found {c} times' )

        return a_copy, b_copy
```

3 Load and display image

- Show images of Lenna and Cameraman

[3]:
```
lenna = 'images/lenna.jpg'
cameraman = 'images/cameraman.jpg'

asg = Assignment(lenna,cameraman)
asg.show_images(plt.imread(lenna),plt.imread(cameraman))
```



4 Assignment Questions:

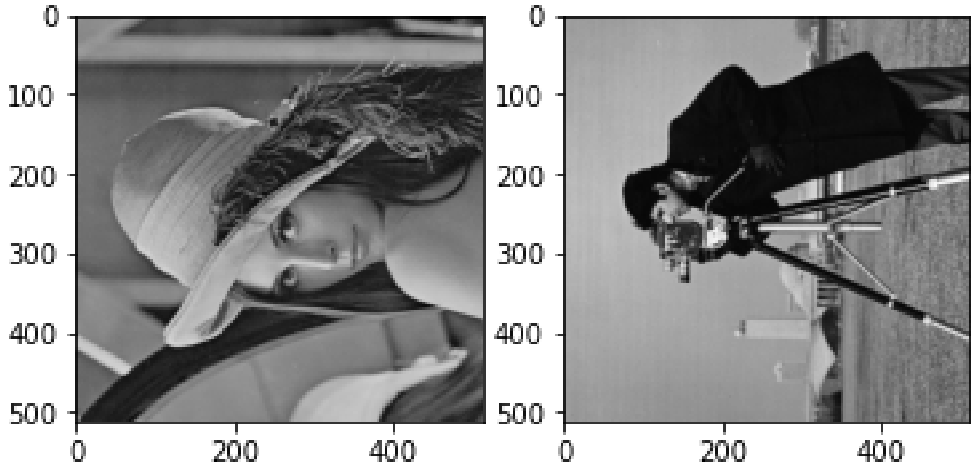
Perform the following operations on the specified images

4.1 Question 3.a : Transpose of images

[4]:

```
start = time.time()
a,b = asg.transpose()
print(f'Execution time: {time.time()-start:.6f} sec')
asg.show_images(a,b)
```

```
Execution time: 0.000000 sec
```

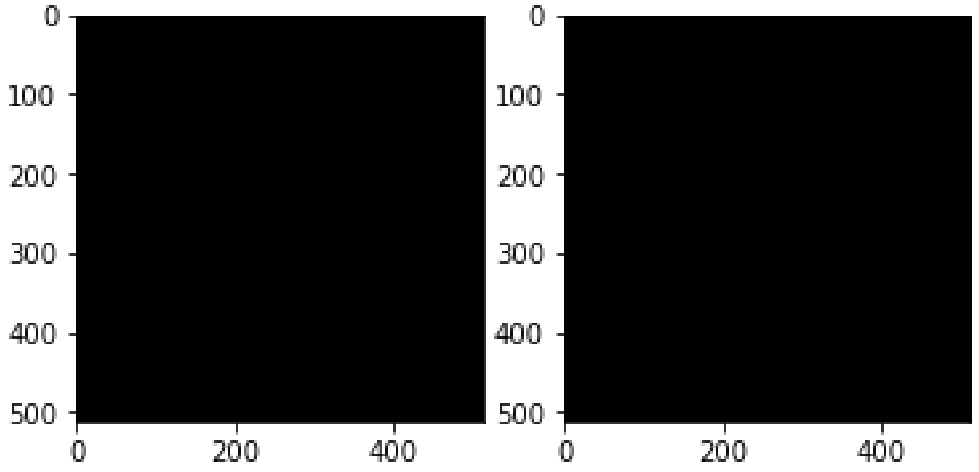


4.2 Question 3.b : Inverse of images

[5]:

```
start = time.time()
a,b = asg.inverse()
print(f'Execution time: {time.time()-start:.6f} sec')
asg.show_images(a,b)
```

```
Execution time: 0.149996 sec
```



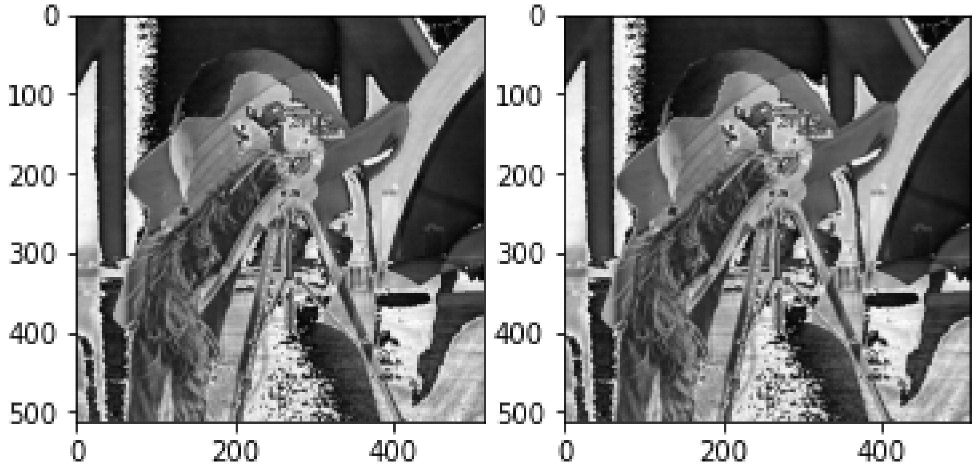
5 Question 3.c : Addition of images (both)

A+B = B+A, therefore both images are same

[6]:

```
start = time.time()
a,b = asg.add()
print(f'Execution time: {time.time()-start:.6f} sec')
asg.show_images(a,b)
```

```
Execution time: 0.002001 sec
```

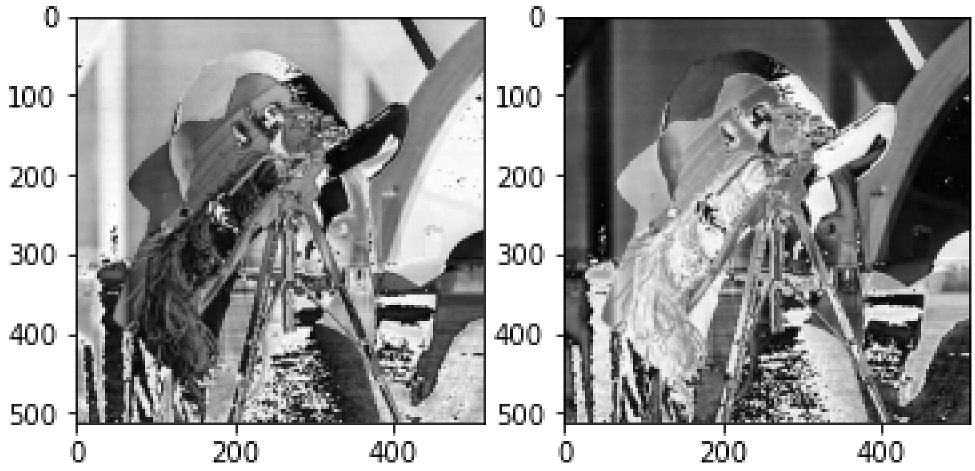


5.1 Question 3.d : Subtraction of images (both)

subtraction doesnt follow commutative property hence they are different

[7]:
```
start = time.time()
a,b = asg.sub()
print(f'Execution time: {time.time()-start:.6f} sec')
asg.show_images(a,b)
```

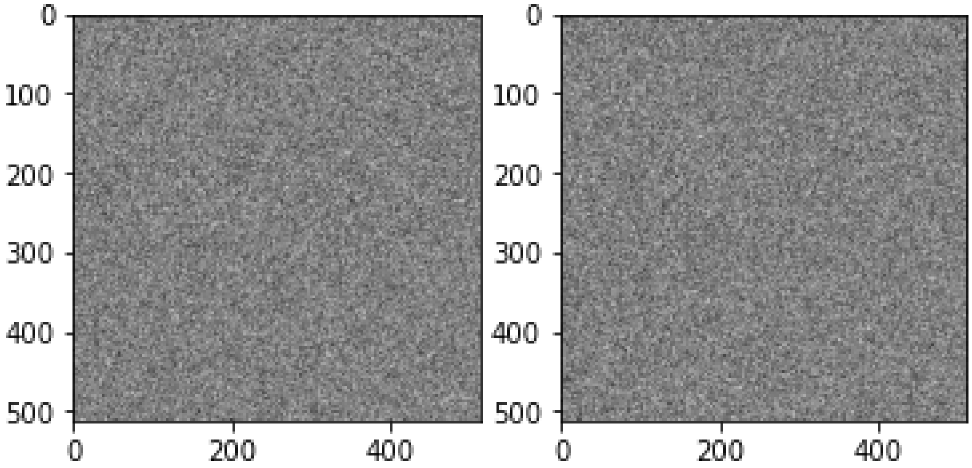
```
Execution time: 0.005003 sec
```



5.2 Question 3.e : Multiplication of images

[8]:
```
start = time.time()
a,b = asg.mul()
print(f'Execution time: {time.time()-start:.6f} sec')
asg.show_images(a,b)
```

```
Execution time: 0.663994 sec
```

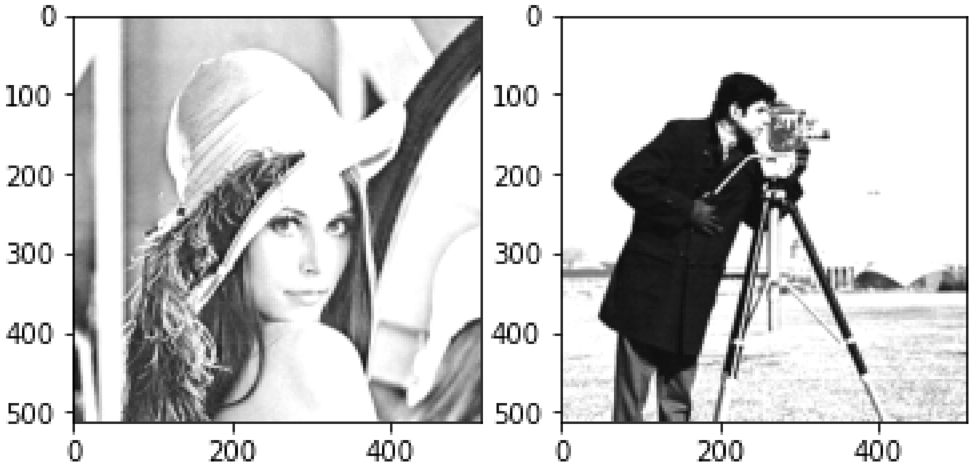


5.3 Question 3.f 2.g: Multiplication of images by scalar greater and less than 1

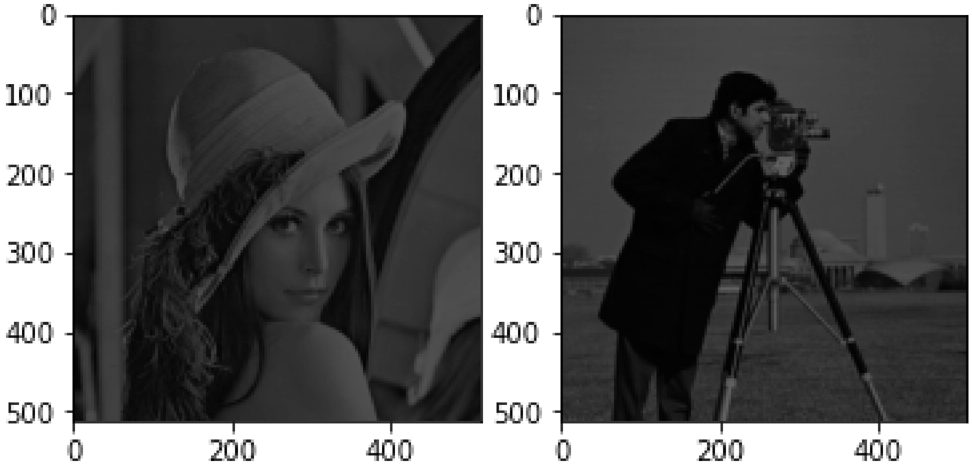
We see that - when the scalar is greater than one, it results in brightening of image - when scalar is less than one, it results in darkening of image

```
[9]: start = time.time()
     a,b = asg.mul_scalar(2)
     print(f'Execution time: {time.time()-start:.6f} sec')
     asg.show_images(a,b)

     start = time.time()
     a,b = asg.mul_scalar(0.4)
     print(f'Execution time: {time.time()-start:.6f} sec')
     asg.show_images(a,b)
```

```
Execution time: 0.007001 sec
```



```
Execution time: 0.036002 sec
```

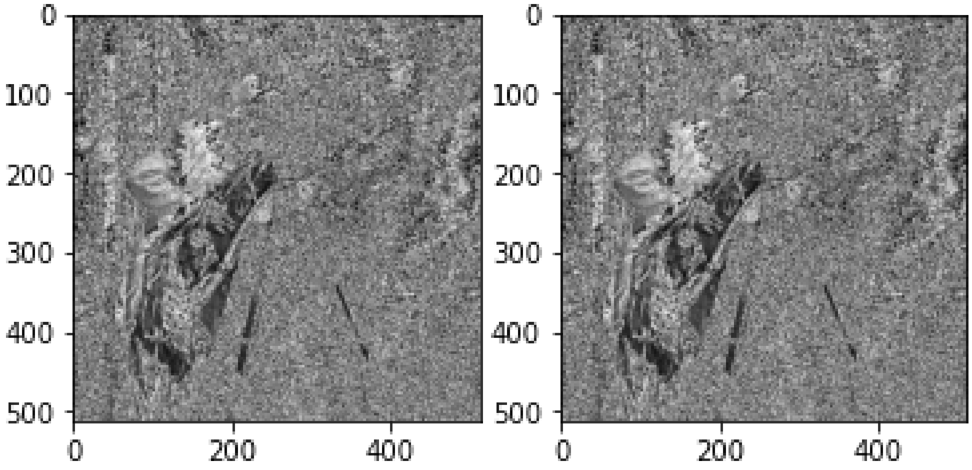


5.4 Question 2.h : Element by Element multiplication

We see that elementwise multiplication is also commutative

[10]:
```python
start = time.time()
a,b = asg.elementwise_mul()
print(f'Execution time: {time.time()-start:.6f} sec')
asg.show_images(a,b)
```

```
Execution time: 0.001001 sec
```



5.5 Question 2.i 2.j : Find the specific value of X

[11]:

```
start = time.time()
asg.value_finder(100)
print(f'Execution time: {time.time()-start:.6f} sec')
```

```
In matrix a, found 100 at : [0,433]
In matrix b, found 100 at : [87,256]
Execution time: 0.005000 sec
```

5.6 Question 2.k : Find the specific value of X and replace it

[14]:

```
start = time.time()
replace_by = 200
# a,b = asg.find_and_replace(0,replace_by)
# print(f'Execution time: {time.time()-start:.6f} sec')
# asg.show_images(a,b)
```

5.7 Question 2.l : Do operations mentioned in assignment

[49]:

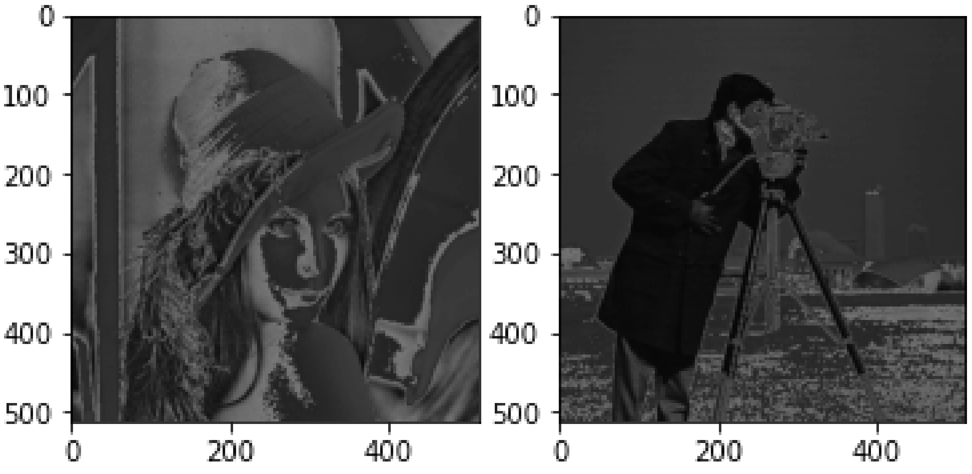
```
class Operations(Assignment):

    def __init__(self,a,b):
        super().__init__(a,b)

    def greater_replace(self, key):
        a_copy = self.a.copy()
        b_copy = self.b.copy()
        (r,c) = a_copy.shape
        for i in range(r):
            for j in range(c):
                if a_copy[i][j] > key:
                    a_copy[i][j] = np.uint8(a_copy[i][j] * 0.3)

        (r,c) = b_copy.shape
        for i in range(r):
            for j in range(c):
                if b_copy[i][j] > key:
                    b_copy[i][j] = np.uint8(b_copy[i][j] * 0.3)

        return a_copy,b_copy

    def less_replace(self, key):
        a_copy = self.a.copy()
        b_copy = self.b.copy()
        (r,c) = a_copy.shape
        for i in range(r):
            for j in range(c):
                if a_copy[i][j]< key:
                    a_copy[i][j] = np.uint8(a_copy[i][j] * 0.3)

        (r,c) = b_copy.shape
        for i in range(r):
            for j in range(c):
                if b_copy[i][j] < key:
                    b_copy[i][j] = np.uint8(b_copy[i][j] * 0.3)
        return a_copy,b_copy

    def greater_less_replace(self):
        a_copy = self.a.copy()
        b_copy = self.b.copy()
        (r,c) = a_copy.shape
        for i in range(r):
            for j in range(c):
                if a_copy[i][j]< 128:
                    a_copy[i][j] = np.uint8(a_copy[i][j] * 0.7)
                else:
                    a_copy[i][j] = np.uint8(a_copy[i][j] * 0.3)

        (r,c) = b_copy.shape
        for i in range(r):
            for j in range(c):
                if b_copy[i][j] < 128:
                    b_copy[i][j] = np.uint8(b_copy[i][j] * 0.7)
                else:
                    b_copy[i][j] = np.uint8(b_copy[i][j] * 0.3)
        return a_copy,b_copy

    def custom_op(self,x):
        a_copy = self.a.copy()
        b_copy = self.b.copy()
        (r,c) = a_copy.shape
        for i in range(r):
            for j in range(c):
                if a_copy[i][j]> 128:
                    a_copy[i][j] = np.uint8(a_copy[i][j] * (0.3 * x + 2))
                else:
                    a_copy[i][j] = np.uint8(a_copy[i][j] * (0.3 * x - 2))

        (r,c) = b_copy.shape
        for i in range(r):
            for j in range(c):
                if b_copy[i][j]> 128:
                    b_copy[i][j] = np.uint8(b_copy[i][j] * (0.3 * x + 2))
                else:
                    b_copy[i][j] = np.uint8(b_copy[i][j] * (0.3 * x - 2))
        return a_copy,b_copy
```

5.7.1 2.l.a : Multiply the intensity values with a constant 0.3 if the intensity value is greater than 127.

[35]:

```
start = time.time()
op = Operations(lenna,cameraman)
a , b = op.greater_replace(120)
print(f'Execution time: {time.time()-start:.6f} sec')
asg.show_images(a,b)
```

```
Execution time: 5.766998 sec
```

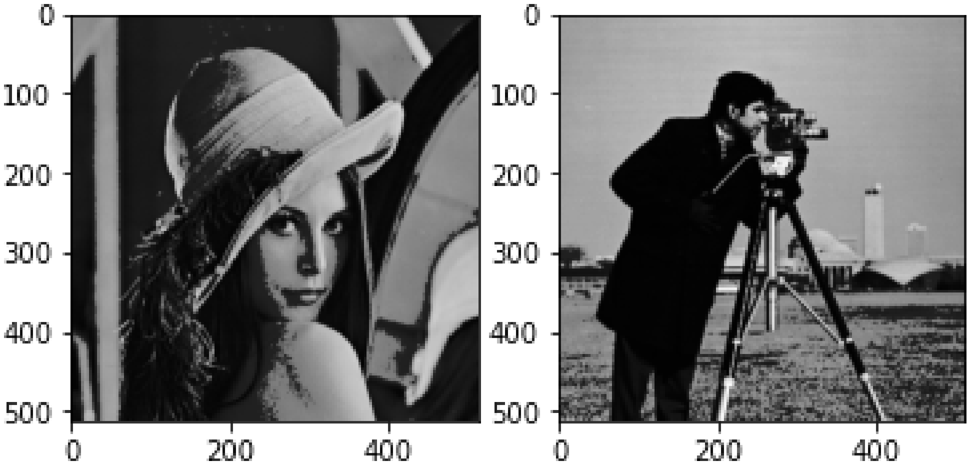


5.7.2 2.l.b : Multiply the intensity values with a constant 0.3 if the intensity value is less than 127.

[42]:

```
start = time.time()
op = Operations(lenna,cameraman)
a , b = op.less_replace(120)
print(f'Execution time: {time.time()-start:.6f} sec')
asg.show_images(a,b)
```

```
Execution time: 5.415994 sec
```

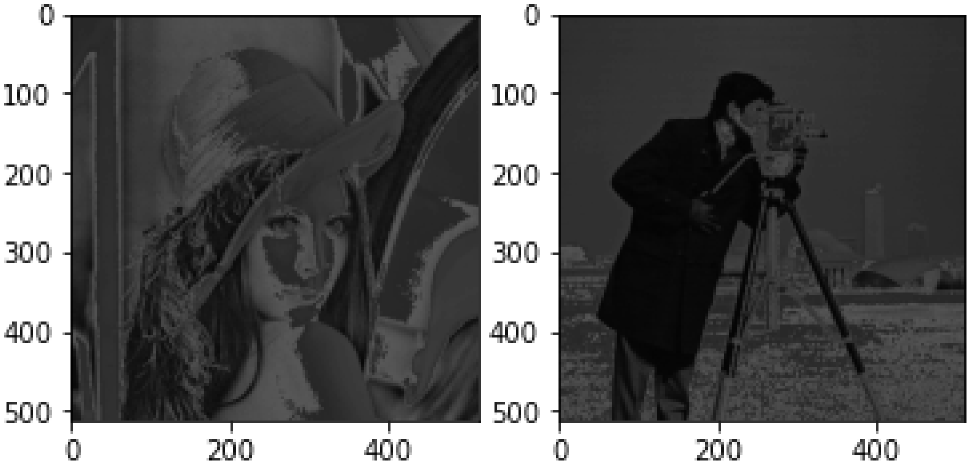


5.7.3 2.l.c : Multiply the intensity values with a constant 0.3 if the intensity value is greater than 127 and with a constant 0.7 if it is less than 128.

[48]:

```
start = time.time()
op = Operations(lenna,cameraman)
a , b = op.greater_less_replace()
print(f'Execution time: {time.time()-start:.6f} sec')
asg.show_images(a,b)
```

```
Execution time: 8.472996 sec
```

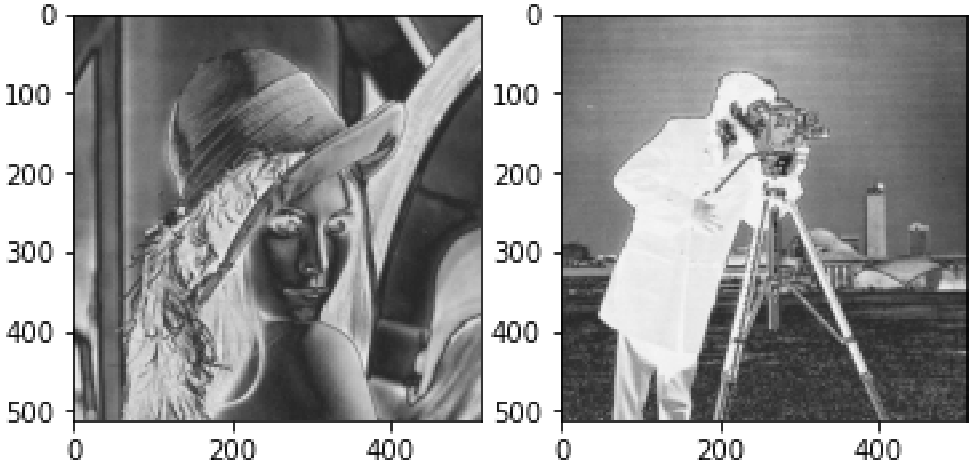


5.7.4 2.l.d : Multiply the intensity values with a equation E_1 if the intensity value is greater than 127 and with a equation E_2 if it is less than 128. $E_1 = 0.3x + 2$; x can take value as $x = 1, 2, 3$. Show and compare the results. $E_1 = 0.3x - 2$; x can take value as $x = 1, 2, 3$. Show and compare the results

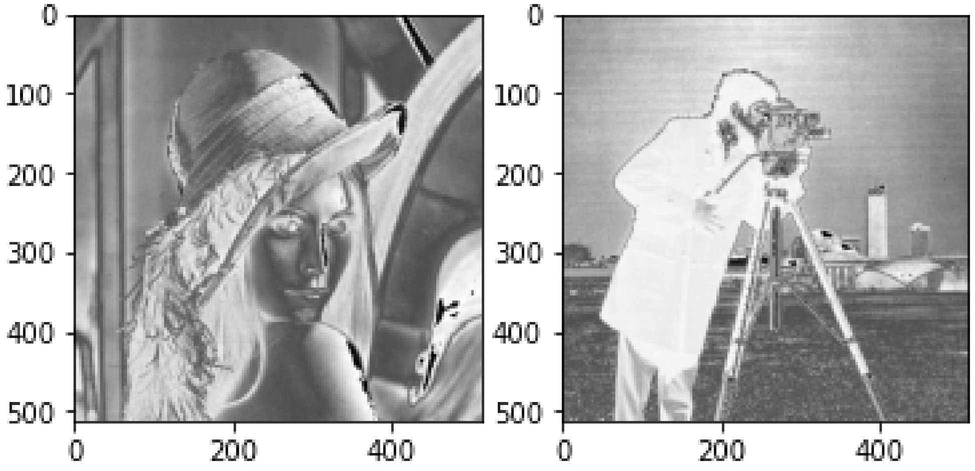
[51]:

```
print('For x = 1')
start = time.time()
op = Operations(lenna,cameraman)
a , b = op.custom_op(1)
print(f'Execution time: {time.time()-start:.6f} sec')
asg.show_images(a,b)

print('For x = 2')
start = time.time()
op = Operations(lenna,cameraman)
a , b = op.custom_op(2)
print(f'Execution time: {time.time()-start:.6f} sec')
asg.show_images(a,b)

print('For x = 3')
start = time.time()
op = Operations(lenna,cameraman)
a , b = op.custom_op(3)
print(f'Execution time: {time.time()-start:.6f} sec')
asg.show_images(a,b)
```

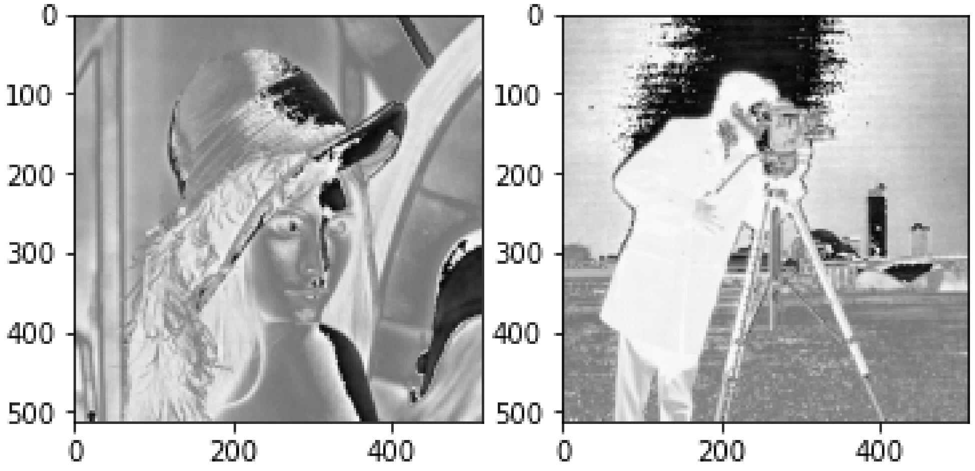
```
For x = 1
Execution time: 8.762995 sec
```



```
For x = 2
Execution time: 8.497997 sec
```



```
For x = 3
Execution time: 8.606997 sec
```



```
[ ]:
```