Q2) Mark four neighbors and eight neighbors of any pixel in the image, implement distance formula, implement image negation, log transformation, and power log Transformation

In [12]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
img=cv2.imread("black.png")
img=cv2.cvtColor(img,cv2.COLOR_BGR2RGB)
```

In [13]:
```
def four_neighbor(img,x,y):
    print("four neighbors with pixel vaues : ")
    print("(",x+1,",",y,") : ",img[x+1,y])
    print("(",x-1,",",y,") : ",img[x+1,y])
    print("(",x,",",y-1,") : ",img[x,y-1])
    print("(",x,",",y+1,") : ",img[x,y+1])

def eight_neighbor(img,x,y):
    print("eight neighbors with pixel vaues : ")
    print("(",x+1,",",y,") : ",img[x+1,y])
    print("(",x-1,",",y,") : ",img[x+1,y])
    print("(",x,",",y-1,") : ",img[x,y-1])
    print("(",x,",",y+1,") : ",img[x,y+1])
    print("(",x+1,",",y+1,") : ",img[x+1,y+1])
    print("(",x-1,",",y-1,") : ",img[x-1,y-1])
    print("(",x+1,",",y-1,") : ",img[x+1,y-1])
    print("(",x-1,",",y+1,") : ",img[x-1,y+1])
```

In [14]:
```
four_neighbor(img,10,20)
```

```
four neighbors with pixel vaues :
( 11 , 20 ) :  [6 6 6]
( 9 , 20 ) :  [6 6 6]
( 10 , 19 ) :  [6 6 6]
( 10 , 21 ) :  [6 6 6]
```

In [15]:
```
eight_neighbor(img,10,20)
```

```
eight neighbors with pixel vaues :
( 11 , 20 ) :  [6 6 6]
( 9 , 20 ) :  [6 6 6]
( 10 , 19 ) :  [6 6 6]
( 10 , 21 ) :  [6 6 6]
( 11 , 21 ) :  [6 6 6]
( 9 , 19 ) :  [6 6 6]
( 11 , 19 ) :  [6 6 6]
( 9 , 21 ) :  [6 6 6]
```

In [ ]:
```

```

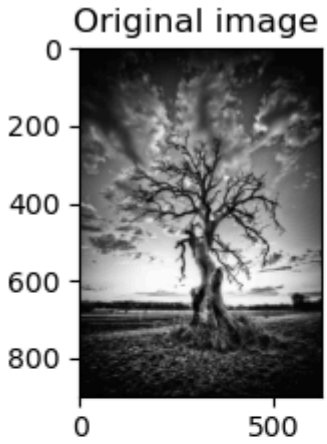
In [16]:
```
def distance(p1,p2):
    d=0
    for i in range(len(p1)):
        d+=(p1[i]-p2[i])**2
    print("Distance : ",np.round(np.sqrt(d),4))
```

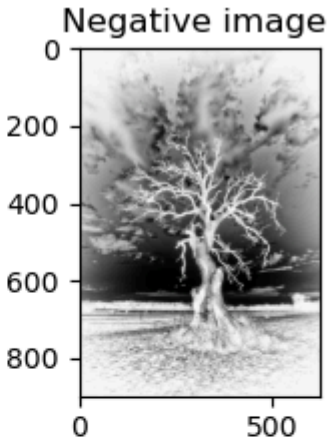
```
p1=(2,3)
p2=(6,7)
distance(p1,p2)
```

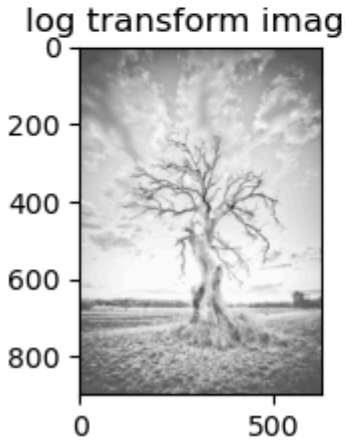
```
Distance :  5.6569
```

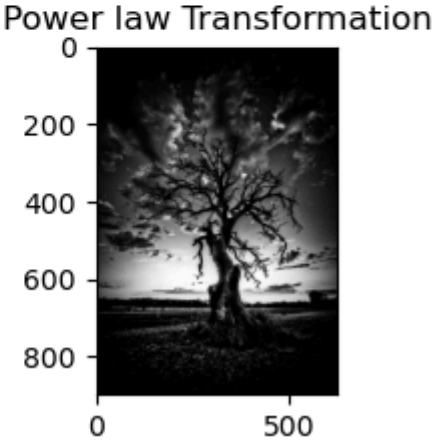
In [18]:

```
img1=cv2.imread("black.png")
img1=cv2.cvtColor(img1,cv2.COLOR_BGR2RGB)


plt.subplot(221)
plt.title("Original image ")
plt.imshow(img1)


plt.subplot(222)
plt.title("Negative image ")
img_negative=255-img1
img_neagtive=cv2.cvtColor(img_negative,cv2.COLOR_BGR2RGB)
plt.imshow(img_neagtive)


plt.subplot(223)
plt.title("log transform image ")
c=255/np.log(2+np.max(img))
img_log=c*np.log1p(img1)
img_log=np.array(img_log, dtype = np.uint8)
img_log=cv2.cvtColor(img_log,cv2.COLOR_BGR2RGB)
plt.imshow(img_log)


plt.subplot(224)
plt.title("Power law Transformation")
gamma=2.2
gamma_corrected = np.array(255*(img1 / 255) ** gamma, dtype = 'uint8')
plt.imshow(gamma_corrected)

plt.tight_layout()
plt.show()
```


Original image
0
200
400
600
800
0
500


Negative image
0
200
400
600
800
0
500


log transform image
0
200
400
600
800
0
500


Power law Transformation
0
200
400
600
800
0
500

In [ ]: