Q.5 : read an image and call it, I_garden. Find the Fourier Transform of I_garden; call it F_garden. Do the following 2 steps :

(a) Create a new spectrum, F_50, where you choose to retain only 50% of the spectrum co-efficients in F_garden, centered around the origin. Zero out the remaining. Use F_50 to reconstruct image I_50.

(b) Create a new spectrum, F_25, where you choose to retain only 25% of the spectrum co-efficients in F_garden, centered around the origin. Zero out the remaining. Use F_25 to reconstruct image I_25.

Compare images I_50 and I_25, with respect to I_garden. What do you observe ?

In [1]:

```
# importing necessary packages

import numpy as np
import matplotlib.pyplot as plt
import cv2
%matplotlib inline
```

Defining a function for fourier transform of an image

In [2]:

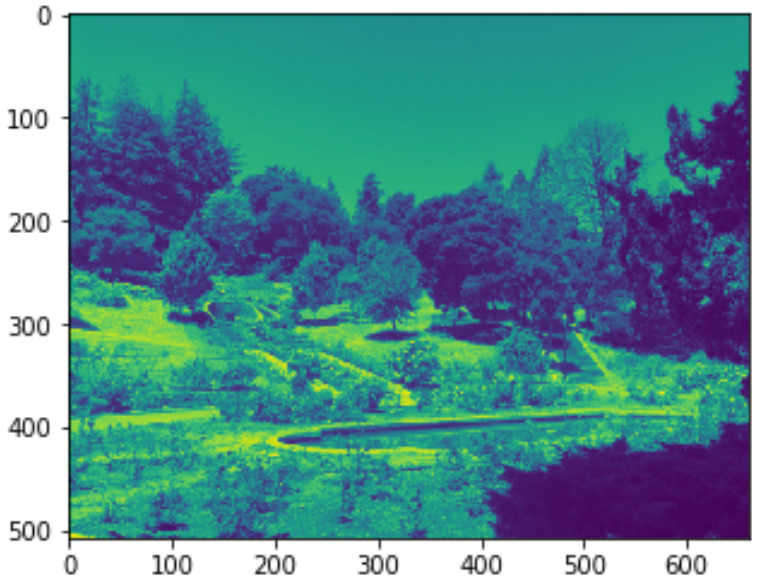
```
def fft(img):
    fft = np.fft.fft2(img)
    fft_shift = np.fft.fftshift(fft)

    return fft_shift

def mag_ph(fft_img):
    fft_shift = fft_img
    magnitude_spectrum = 20*np.log(0.00001+np.abs(fft_shift))
    phase_spectrum = np.angle(fft_shift)

    return magnitude_spectrum, phase_spectrum
```

1. Reading an image into I_garden

In [3]:

```
I_garden = cv2.imread('./images/5.png',0)
plt.imshow(I_garden)
print(I_garden.shape)
```

```
(508, 660)
```



2. Taking Fourier transform of an image

In [4]:

```
F_garden = fft(I_garden)
F_garden.shape
```

Out[4]:

```
(508, 660)
```

Flow of steps :

First I will create a mask / low pass filter for an image , which will allow low pass frequncies to pass out and zero out the higher frequency. In fftshift of an image low frequencies are around DC component of an image : which is at the middle.So we create a filter accordingly

3. creating a mask / low pass filter : L_50 : which will allow 50% of spec coeff to pass around an origin

In [5]:

```
# taking size of an image to create a mask
ncols = int(660)
nrows = int(508)

#print(type(nrows))

# 50% mask
L_50 = np.zeros((nrows,ncols))
L_50[int(nrows/2-nrows/4) : int(nrows/2+nrows/4) , int(ncols/2-nc
```

4. Applying low pass filter / 50% Mask / L_50 to fourier transformed image F_garden

In [6]:

```
F_50 = F_garden * L_50
```

5. recovering image frm fft by tkng inv fftshift first then tkng inv fft

In [7]:

```
x = np.fft.ifftshift(F_50)
y = np.fft.ifft2(x)
F_50_img = np.array(np.abs(y))          # np.abs removes imaginar

```

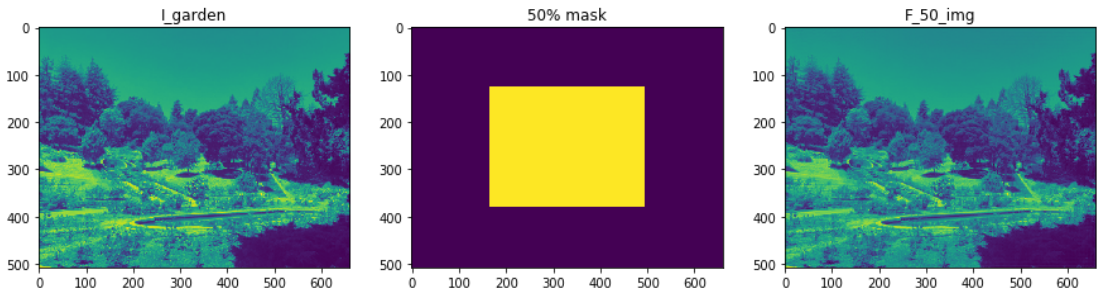
6. Plotting an original image + Mask representation + Image recovered after applying low pass filter

In [8]:

```
f, axarr = plt.subplots(1,3,figsize=(15,15))
axarr[0].imshow(I_garden); axarr[0].set_title('I_garden')
axarr[1].imshow(L_50) ; axarr[1].set_title('50% mask')
axarr[2].imshow(F_50_img) ; axarr[2].set_title('F_50_img')

```

Out[8]:

```
Text(0.5,1,'F_50_img')
```



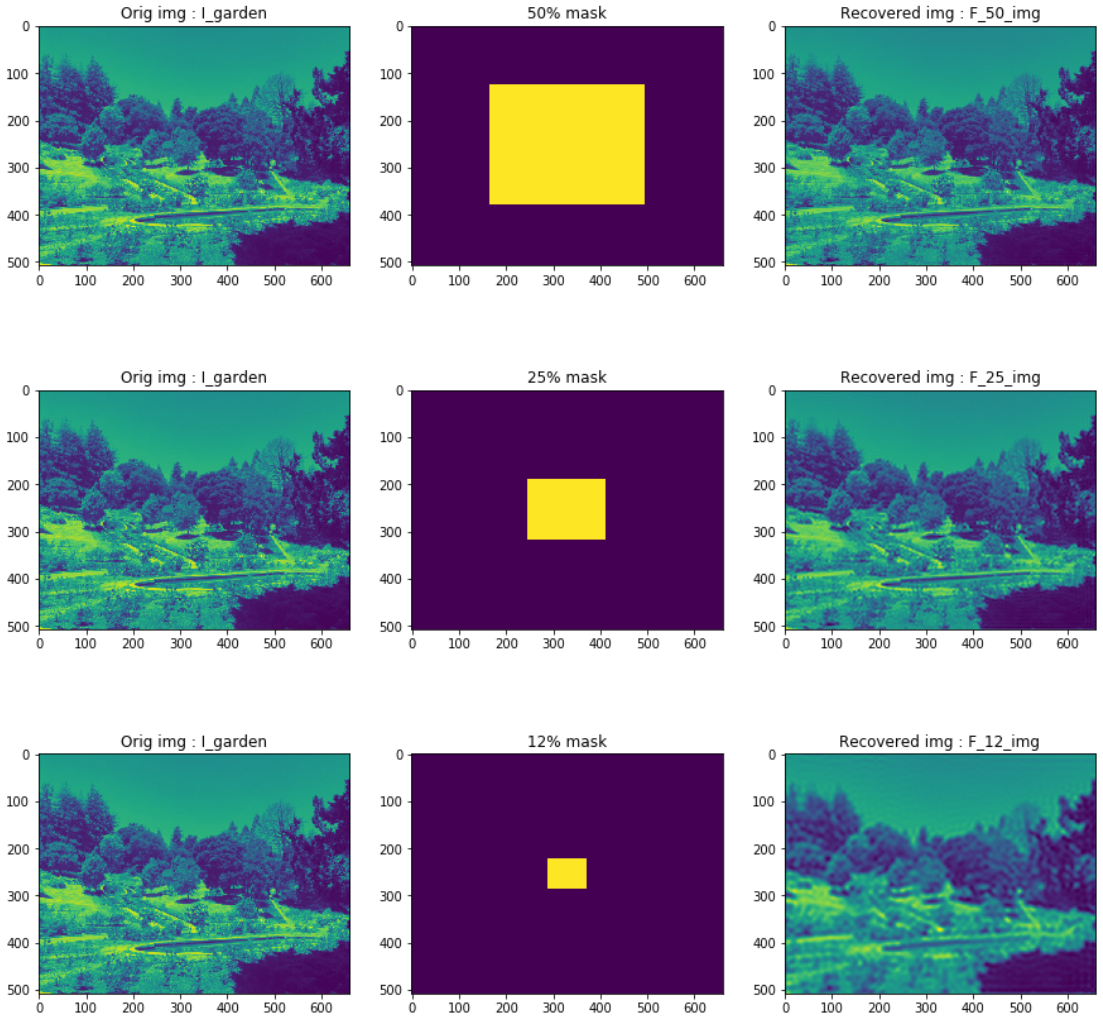
7. Likewise , follwing similar steps for two more masks / low pass filters : 25% and 12%

In [9]:

```
# 25% mask
L_25 = np.zeros((nrows,ncols))
L_25[int(nrows/2-nrows/8) : int(nrows/2+nrows/8) , int(ncols/2-nc

# 12.5% mask
L_12 = np.zeros((nrows,ncols))
L_12[int(nrows/2-nrows/16) : int(nrows/2+nrows/16) , int(ncols/2-

# applying low pass filter to fourier transformed img
F_25 = F_garden * L_25
F_12 = F_garden * L_12

# recovering image frm fft by tkng inv fftshift first then tkng i
x = np.fft.ifftshift(F_25)
y = np.fft.ifft2(x)
F_25_img = np.array(np.abs(y))
x = np.fft.ifftshift(F_12)
y = np.fft.ifft2(x)
F_12_img = np.array(np.abs(y))

# plotting all the results
f, axarr = plt.subplots(3,3,figsize=(15,15))
axarr[0,0].imshow(I_garden); axarr[0,0].set_title('Orig img : I_g
axarr[0,1].imshow(L_50) ; axarr[0,1].set_title('50% mask')
axarr[0,2].imshow(F_50_img) ; axarr[0,2].set_title('Recovered img
axarr[1,0].imshow(I_garden) ; axarr[1,0].set_title('Orig img : I_
axarr[1,1].imshow(L_25) ; axarr[1,1].set_title('25% mask')
axarr[1,2].imshow(F_25_img) ; axarr[1,2].set_title('Recovered img
axarr[2,0].imshow(I_garden) ; axarr[2,0].set_title('Orig img : I_
axarr[2,1].imshow(L_12) ; axarr[2,1].set_title('12% mask')
axarr[2,2].imshow(F_12_img) ; axarr[2,2].set_title('Recovered img
```

Out[9]:

```
Text(0.5,1,'Recovered img : F_12_img')
```



Observation :

- As we can see after applying a mask / low pass filter , it allows only low frequency components or spectrum - -coefficeints around an origin (DC component) : This will in turn reduce an information or details of an image: Which results into a blur image with low amount of details
- As we are reducing the size of the mask / low pass filter : it is cutting off more frequencies that are high than given threshold or distant from center in this case : so it results into more blur image and less info

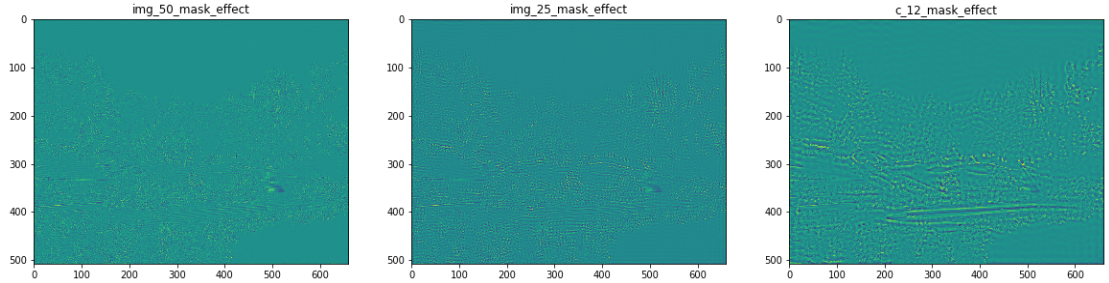
8. Looking at the effect of particular mask on an image (by removing recoverd fourier image from the original image)

In [15]:

```
a = I_garden - F_50_img
b =I_garden - F_25_img
c = I_garden - F_12_img

f, axarr = plt.subplots(1,3,figsize=(20,20))
axarr[0].imshow(a); axarr[0].set_title('img_50_mask_effect')
axarr[1].imshow(b) ; axarr[1].set_title('img_25_mask_effect')
axarr[2].imshow(c) ; axarr[2].set_title('c_12_mask_effect')
```

Out[15]:

```
Text(0.5,1,'c_12_mask_effect')
```



Observation :

- Blurring (smoothing / removal of fine or sharp details (edges)) is happening in the
image except the sky region
- if the mask is too low such as 12% then sky is blurred a bit
- Ringing effects are there in regions where we have sharp edges in the original image (Edges are the high frequency components in fourier transformed image , because of their behaviour of sudden change , and as we are applying the low pass filter , it won't allow sharp edges to pass)

- The main cause of ringing effect is due to a signal being bandlimited (specifically, not having high frequencies) or passed through a ideal low-pass filter; this is the frequency domain description.

- solution is Butterworth low pass filter that attenuates high frequencies smoothly