```python
import cv2 as cv
import matplotlib.pyplot as plt
import numpy as np
import math

# path to the input img
path = 'C:/Users/Raiyan/Desktop/building_332x317.jpg'

# reading img + converting from BGR to GRAY
img = cv.imread(path)
img = cv.cvtColor(img, cv.COLOR_BGR2GRAY)

k_h = int(input("Enter kernel height: "))
k_w = k_h
k_size = (k_h,k_w)

# empty kernel
kernel = np.zeros( k_size, np.float32)

# img height
img_h = img.shape[0]
# img width
img_w = img.shape[1]
# kernel height // 2
a = kernel.shape[0] // 2
# kernel width // 2
b = kernel.shape[1] // 2

sigma = 60.0
normalizing_c = 1.0 / ( 2.0 * sigma * sigma * np.pi )

# building gauss kernel
for x in range(-a,a+1):
    for y in range(-b,b+1):
        dist = math.sqrt(x*x + y*y) * normalizing_c
        val = math.exp( -dist ) * normalizing_c
        kernel[a+x][b+y] = val

# empty op img
output = np.zeros((img_h,img_w), np.float32)

# conv
# visiting each pixel in the img
# m ta row img e ... for each row ...
for i in range(img_h):
    # n ta coln img e ... for each coln ...
    for j in range(img_w):
        # sum of val to be calc
        temp = 0
        # visiting each pixel in the kernel
        # a ta row img e ... for each row ...
        for x in range(-a,a+1):
            # b ta coln img e ... for each coln ...
            for y in range(-b,b+1):
                if 0 <= i-x < img_h and 0 <= j-y < img_w:
                    temp += kernel[a+x][b+y] * img[i-x][j-y]
                else:
                    temp += 0
        temp = temp / (k_w*k_h)
        output[i][j] = temp


def show_images(images, image_title):
    # displaying multiple images side by side
    # https://stackoverflow.com/questions/41793931/plotting-images-side-by-side-using-matplotlib

    # err : was giving weird colormap due to diff in the mechanism of reading img of cv2 & matplotlib
    # https://stackoverflow.com/questions/3823752/display-image-as-grayscale-using-matplotlib
    # running this once in the code will ALWAYS give gray op
    plt.gray()

    no_of_imgs = len(images)
    f = plt.figure()
    for i in range(no_of_imgs):
```

```
75
76          # Debug, plot figure
77          axes = f.add_subplot(1, no_of_imgs, i + 1)
78          # the last img will show y axis on the RHS instead of LHS(which is by
    default)
79
80          if i==no_of_imgs-1:
81              axes.yaxis.tick_right()
82
83          plt.title(image_title[i])
84          plt.imshow(images[i], 'gray')
85          # plt.rc('font', size=8)
86      plt.show(block=True)
87
88  show_images([img,output], ['input', 'output'])
89
```