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

def show_images(images):
    # 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()

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

        # Debug, plot figure
        axes = f.add_subplot(1, n, i + 1)
        # the last img will show y axis on the RHS instead of LHS(which is by default)
        if i==n-1:
            axes.yaxis.tick_right()
        plt.imshow(images[i])

    plt.show(block=True)

# path to the input img
# path = "C:/Users/Raiyan/Desktop/img/03/Image-Processing-and-Computer-Vision-Lab/Lab 2/Average filter/Input.png"
path = 'C:/Users/Raiyan/Desktop/building.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)

# avg kernel
kernel = np.ones( k_size, np.float32)

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

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

# sum of the values of the kernel
k_sum = kernel.sum()
# print(f'ksum is {ksum}')

# 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):

        # empty var for calculating a summed value
        value = 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:
                    value = value + kernel[a+x][b+y] * img[i+x][j+y]
                else:
                    value = value + 0
        value = value / k_sum
        output[i][j] = value

show_images([img,output])


```