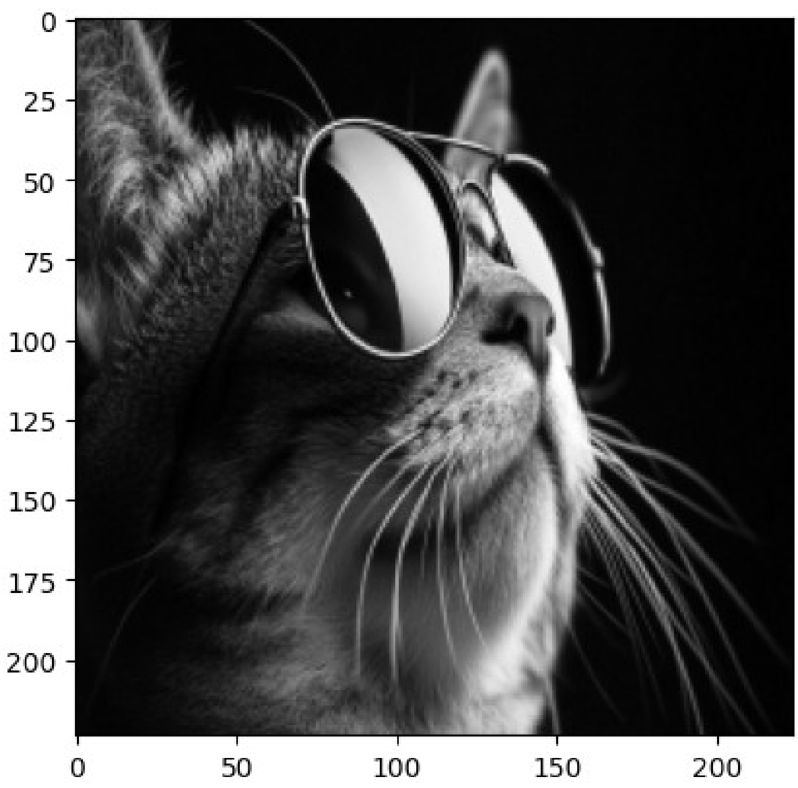
```python
# Importing the libraries
import numpy as np
import matplotlib.pyplot as plt
import cv2
```

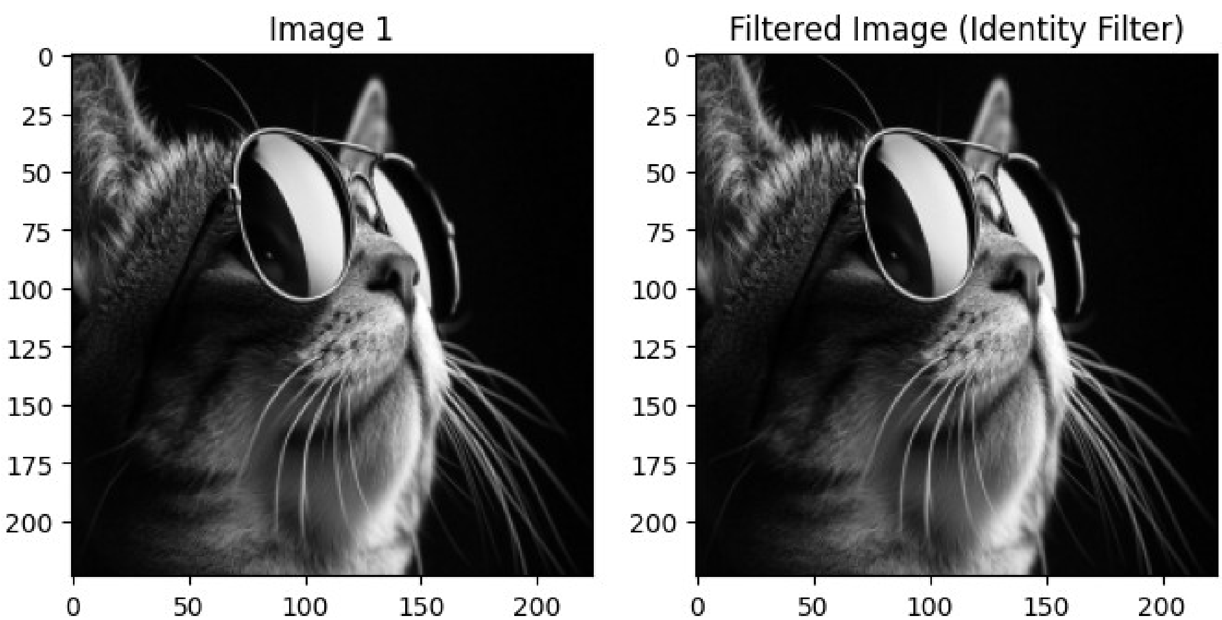
Task 1

```python
image1 = cv2.imread("/content/drive/MyDrive/storage extension/Colab Notebooks/CSE463/Lab 2/23341134_UdoySaha_Lab2/Images/Image1.jpg")
image1 = cv2.resize(image1, (224, 224), interpolation=cv2.INTER_AREA)
grayscale_image1 = cv2.cvtColor(image1, cv2.COLOR_BGR2GRAY)
plt.imshow(grayscale_image1, cmap='gray')
plt.show()
```




```python
def apply_kernel(image, kernel):
  filtered_image = cv2.filter2D(image, -1, kernel)
  return np.clip(filtered_image, 0, 255)

# applying a 3x3 identity kernel
kernel1 = np.array([[0, 0, 0],
                    [0, 1, 0],
```

```python
                    [0, 0, 0]])

filtered_image1 = apply_kernel(grayscale_image1, kernel1)

# Plotting the images
plt.figure(figsize=(8, 4))

plt.subplot(1, 2, 1)
plt.imshow(grayscale_image1, cmap='gray')
plt.title('Image 1')

plt.subplot(1, 2, 2)
plt.imshow(filtered_image1, cmap='gray')
plt.title('Filtered Image (Identity Filter)')

plt.show()
```



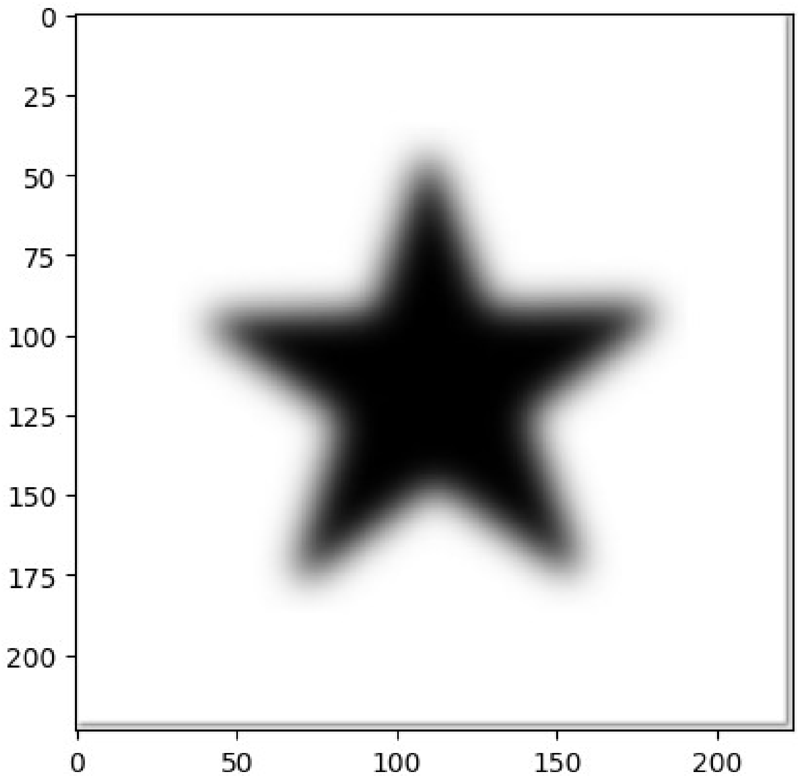
###Explanation: ___

- Since identity filter is designed to retain all the image information and change nothing, this kernel has no impact on the input image to the output image.

- As `cv2.filter2D` automatically pads the input image, the input and output image is completely identical.

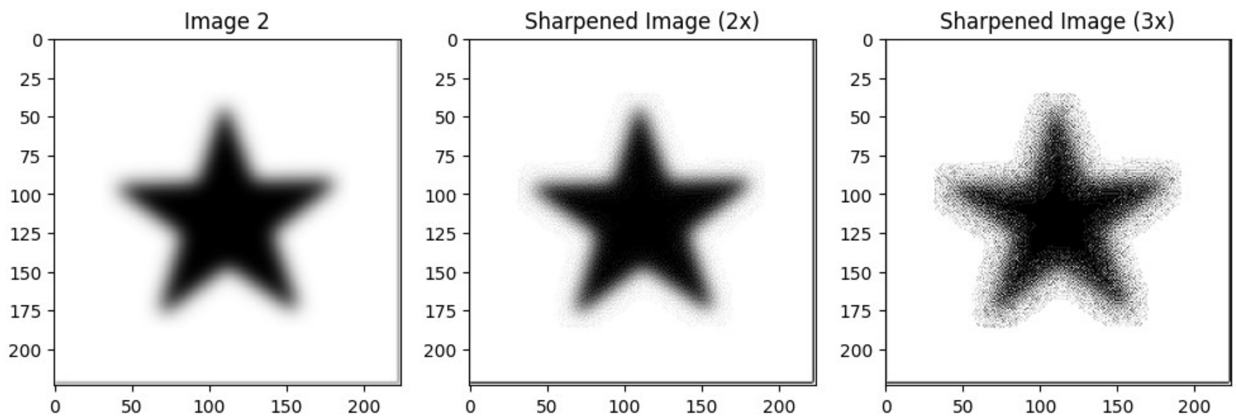
#Task 2

```
image2 = cv2.imread("/content/drive/MyDrive/storage extension/Colab
Notebooks/CSE463/Lab 2/23341134_UdoySaha_Lab2/Images/Image2.jpg")
image2 = cv2.resize(image2, (224, 224), interpolation=cv2.INTER_AREA)
image2_rgb = cv2.cvtColor(image2, cv2.COLOR_BGR2RGB)
plt.imshow(image2_rgb)
plt.show()
```




```
# 3x3 sharpening kernel
kernel2 = np.array([[ 0, -1,  0],
                    [-1,  5, -1],
                    [ 0, -1,  0]])

sharp_image2_1 = apply_kernel(image2_rgb, kernel2)
sharp_image2_1 = apply_kernel(sharp_image2_1, kernel2)

sharp_image2_2 = apply_kernel(sharp_image2_1, kernel2)

# Plotting the images
plt.figure(figsize=(12, 4))

plt.subplot(1, 3, 1)
plt.imshow(image2_rgb)
plt.title('Image 2')
```

```
plt.subplot(1, 3, 2)
plt.imshow(sharp_image2_1)
plt.title('Sharpened Image (2x)')

plt.subplot(1, 3, 3)
plt.imshow(sharp_image2_2)
plt.title('Sharpened Image (3x)')

plt.show()
```



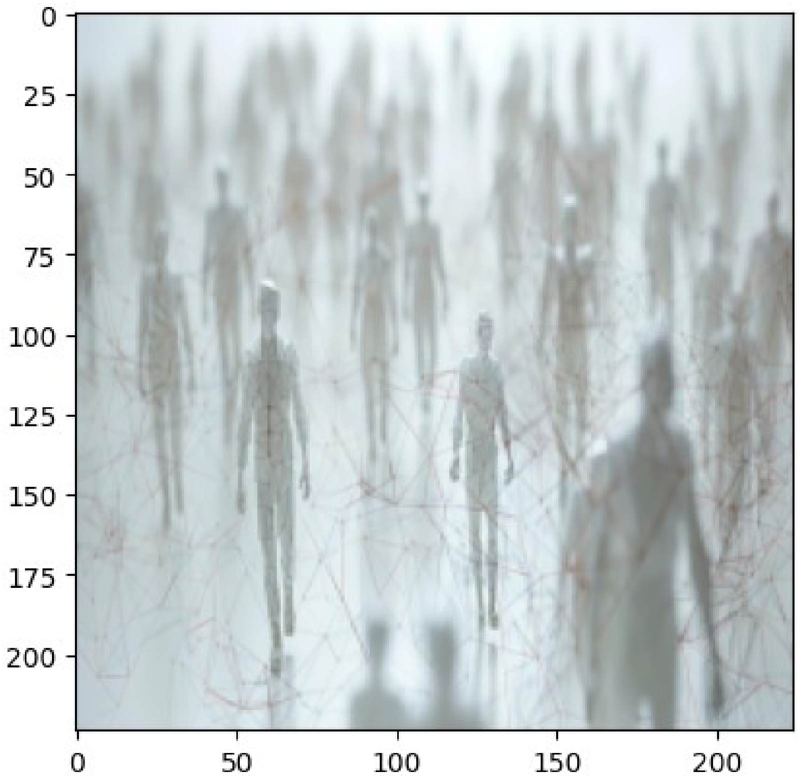
###Explanation: ___

- The sharpening filter shifts the pixel value of the center pixel away from the neigboring pixels.

- Thus, there is a contrast created between the pixel values of neigboring pixels and it becomes easier to identify each object visually, making the picture sharper.

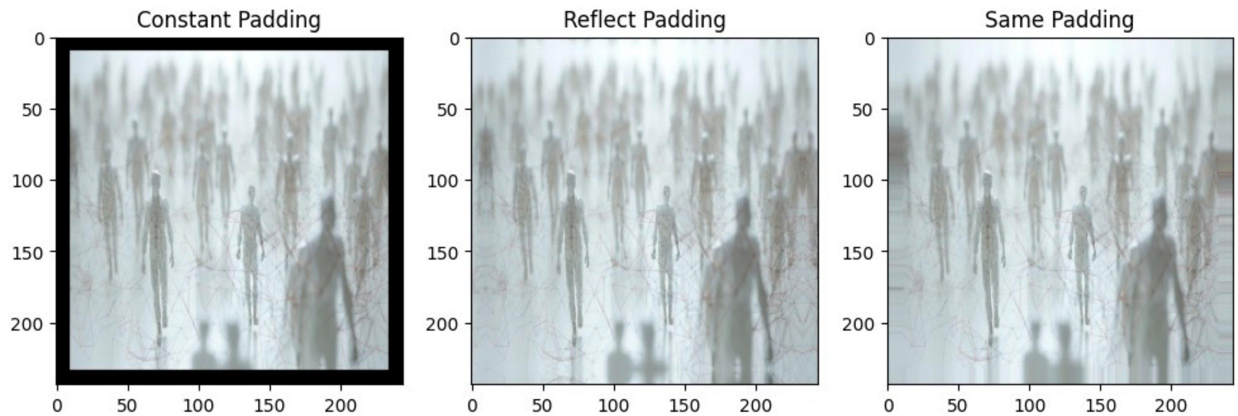
#Task 3

```
image3 = cv2.imread("/content/drive/MyDrive/storage extension/Colab Notebooks/CSE463/Lab 2/23341134_UdoySaha_Lab2/Images/Image3.jpg")
image3 = cv2.resize(image3, (224, 224), interpolation=cv2.INTER_AREA)
image3_rgb = cv2.cvtColor(image3, cv2.COLOR_BGR2RGB)
plt.imshow(image3_rgb)
plt.show()
```

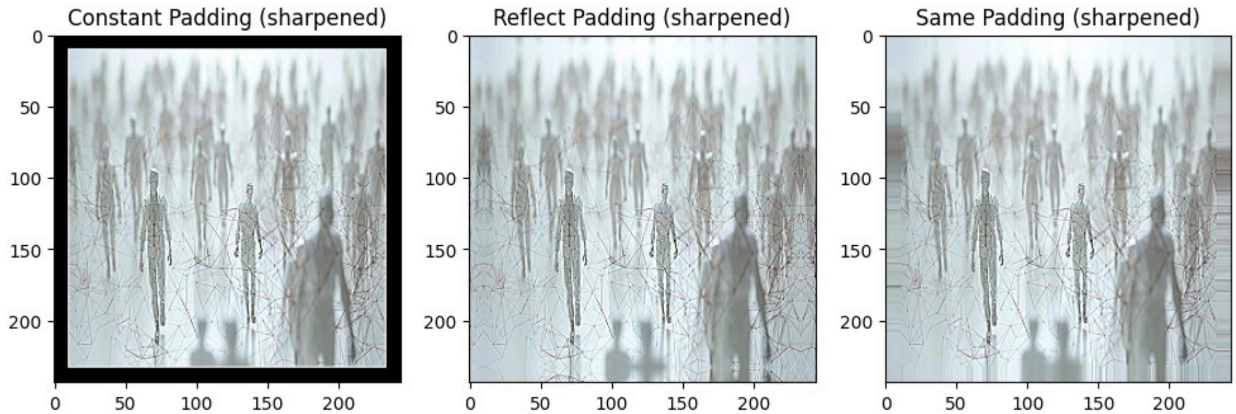



```
pad_wid = 10  # 10 so that we can have a clear view of whats
happening. Otherwise, 1 is great.

# Constant padding with 0
constant_padded_img = np.pad(image3_rgb, ((pad_wid, pad_wid),
(pad_wid, pad_wid), (0, 0)), mode='constant', constant_values=0)

# Reflect padding
reflect_padded_img = np.pad(image3_rgb, ((pad_wid, pad_wid), (pad_wid,
pad_wid), (0, 0)), mode='reflect')

# Same/Edge padding
same_padded_img = np.pad(image3_rgb, ((pad_wid, pad_wid), (pad_wid,
pad_wid), (0, 0)), mode='edge')

# Plotting the images
plt.figure(figsize=(12, 4))

plt.subplot(1, 3, 1)
plt.imshow(constant_padded_img)
plt.title('Constant Padding')

plt.subplot(1, 3, 2)
plt.imshow(reflect_padded_img)
plt.title('Reflect Padding')
```

```
plt.subplot(1, 3, 3)
plt.imshow(same_padded_img)
plt.title('Same Padding')

plt.show()
```

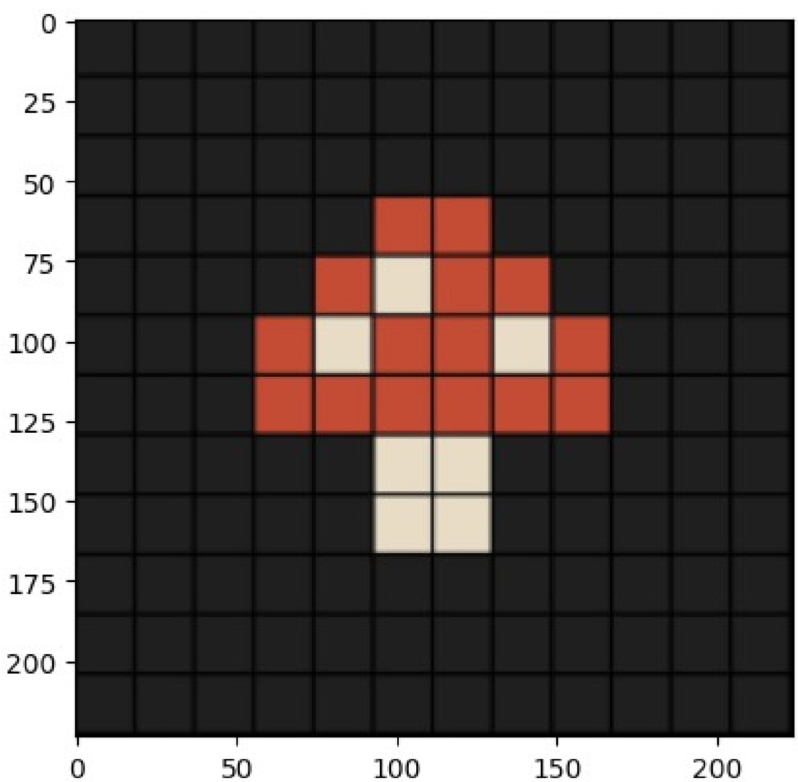



```
sharp_constant_padded_img = apply_kernel(constant_padded_img, kernel2)
sharp_reflect_padded_img = apply_kernel(reflect_padded_img, kernel2)
sharp_same_padded_img = apply_kernel(same_padded_img, kernel2)

# Plotting the images
plt.figure(figsize=(12, 4))

plt.subplot(1, 3, 1)
plt.imshow(sharp_constant_padded_img)
plt.title('Constant Padding (sharpened)')

plt.subplot(1, 3, 2)
plt.imshow(sharp_reflect_padded_img)
plt.title('Reflect Padding (sharpened)')

plt.subplot(1, 3, 3)
plt.imshow(sharp_same_padded_img)
plt.title('Same Padding (sharpened)')

plt.show()
```

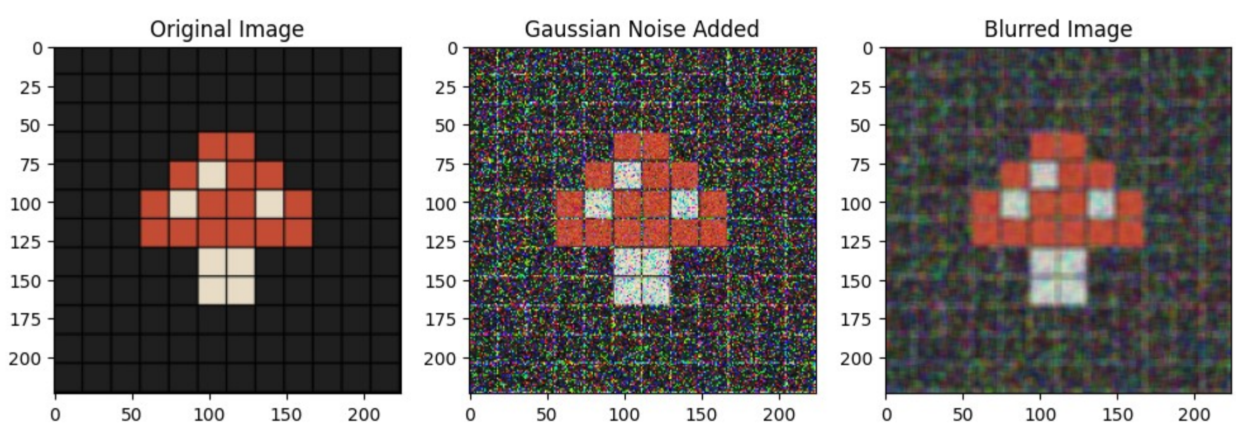


#Task 4

```
image4 = cv2.imread("/content/drive/MyDrive/storage extension/Colab Notebooks/CSE463/Lab 2/23341134_UdoySaha_Lab2/Images/Image4.jpg")
image4 = cv2.resize(image4, (224, 224), interpolation=cv2.INTER_AREA)
image4_rgb = cv2.cvtColor(image4, cv2.COLOR_BGR2RGB)
plt.imshow(image4_rgb)
plt.show()
```



```python
# Function to add Gaussian noise to an image
def add_gaussian_noise(image, mean=0, sigma=25):
    noisy_image = image + np.random.normal(mean, sigma,
image.shape).astype(np.uint8)
    return noisy_image

noisy_image4 = add_gaussian_noise(image4_rgb)

# Applying Average filter
blurred_image4 = cv2.blur(noisy_image4, (5, 5))

# Plotting the images
plt.figure(figsize=(12, 4))

plt.subplot(1, 3, 1)
plt.imshow(image4_rgb)
plt.title('Original Image')

plt.subplot(1, 3, 2)
plt.imshow(noisy_image4)
plt.title('Gaussian Noise Added')

plt.subplot(1, 3, 3)
plt.imshow(blurred_image4)
plt.title('Blurred Image')

plt.show()
```

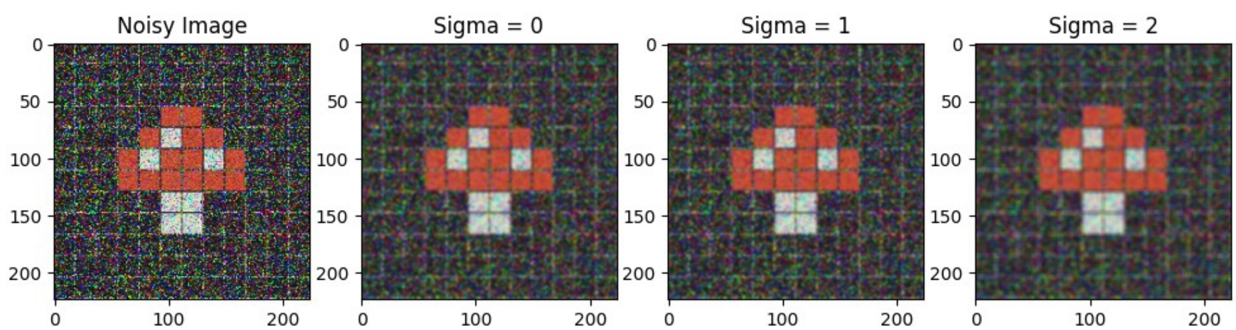


###Explanation: ___

- After applying Average Filter, the noisy image gets smooth overall.
- The noises get reduced, but the sharpness of the image was affected.

Task 5

```
gblur1 = cv2.GaussianBlur(noisy_image4, (5, 5), sigmaX=0)
gblur2 = cv2.GaussianBlur(noisy_image4, (5, 5), sigmaX=1)
gblur3 = cv2.GaussianBlur(noisy_image4, (5, 5), sigmaX=2)

# Plotting the images
plt.figure(figsize=(12, 4))

plt.subplot(1, 4, 1)
plt.imshow(noisy_image4)
plt.title('Noisy Image')

plt.subplot(1, 4, 2)
plt.imshow(gblur1)
plt.title('Sigma = 0')

plt.subplot(1, 4, 3)
plt.imshow(gblur2)
plt.title('Sigma = 1')

plt.subplot(1, 4, 4)
plt.imshow(gblur3)
plt.title('Sigma = 2')

plt.show()
```



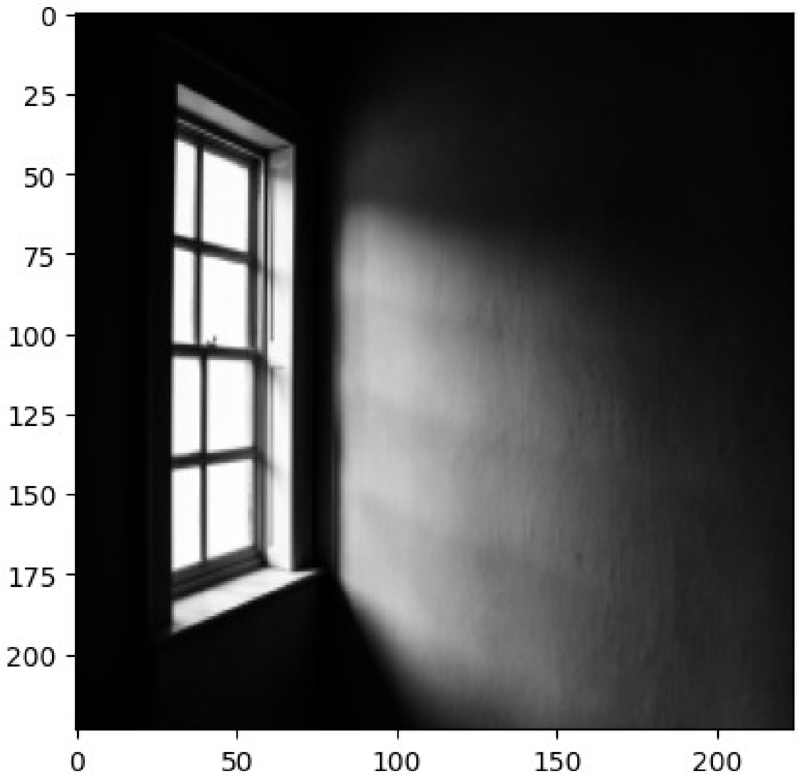
###Explanation: ___

- Smooting is less when the Sigma value is less.
- Noise gets reduced with large Sigma values.

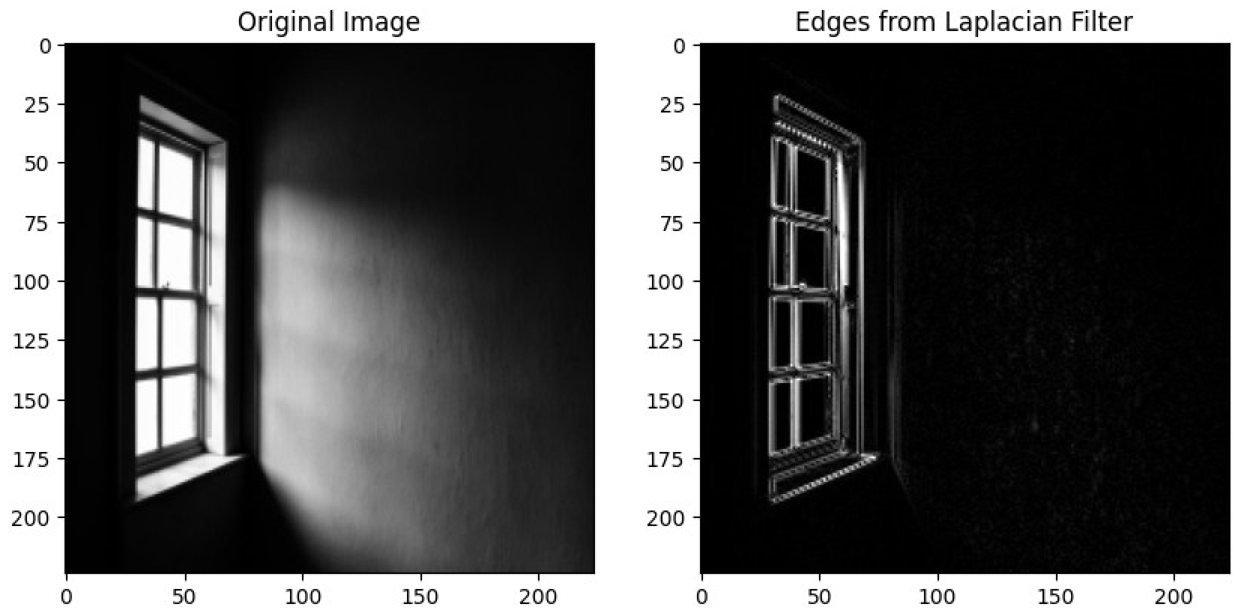
Task 6

```
image5 = cv2.imread("/content/drive/MyDrive/storage extension/Colab Notebooks/CSE463/Lab 2/23341134_UdoySaha_Lab2/Images/Image5.jpg")
```

```
image5 = cv2.resize(image5, (224, 224), interpolation=cv2.INTER_AREA)
grayscale_image5 = cv2.cvtColor(image5, cv2.COLOR_BGR2GRAY)
plt.imshow(grayscale_image5, cmap='gray')
plt.show()
```

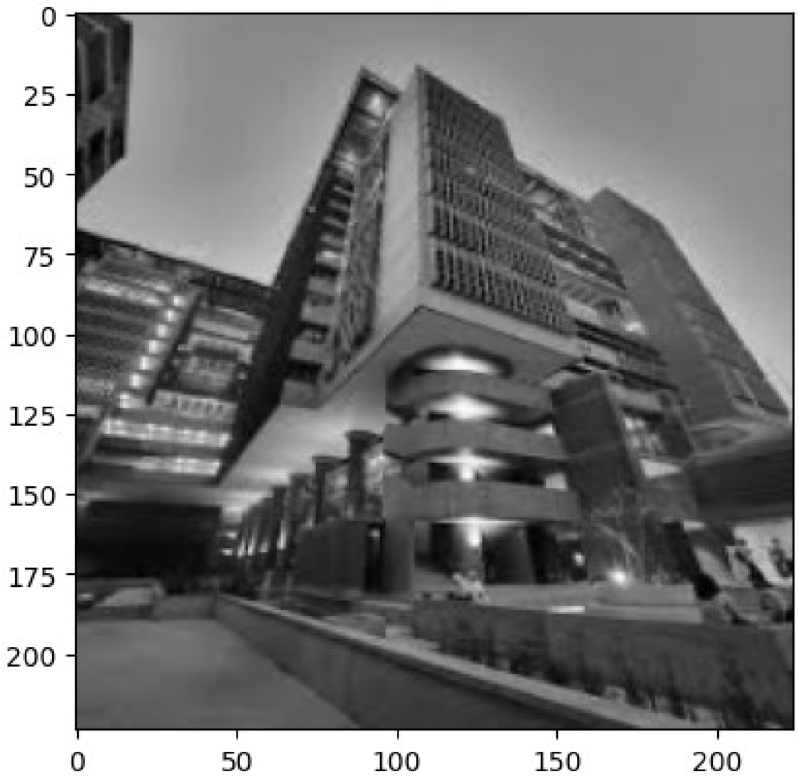



```
# Apply Laplacian filter
edges5 = cv2.Laplacian(grayscale_image5, cv2.CV_64F)  # Use 64F to 
handle negative values
edges5 = cv2.convertScaleAbs(edges5)  # Convert back to 8-bit

# Plotting the images
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.imshow(grayscale_image5, cmap='gray')
plt.title('Original Image')

plt.subplot(1, 2, 2)
plt.imshow(edges5, cmap='gray')
plt.title('Edges from Laplacian Filter')

plt.show()
```



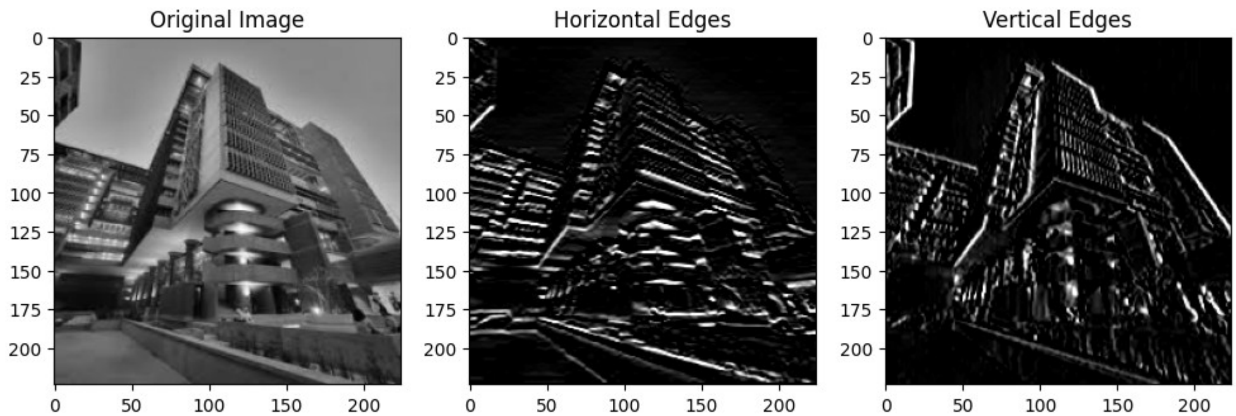
#Task 7

```
image6 = cv2.imread("/content/drive/MyDrive/storage extension/Colab Notebooks/CSE463/Lab 2/23341134_UdoySaha_Lab2/Images/Image6.jpg")
image6 = cv2.resize(image6, (224, 224), interpolation=cv2.INTER_AREA)
grayscale_image6 = cv2.cvtColor(image6, cv2.COLOR_BGR2GRAY)
plt.imshow(grayscale_image6, cmap='gray')
plt.show()
```




```python
kernel_vertical = np.array([[-1, 0, 1],
                            [-2, 0, 2],
                            [-1, 0, 1]])

kernel_horizontal = np.array([[-1, -2, -1],
                              [ 0,  0,  0],
                              [ 1,  2,  1]])

edges_horizontal = apply_kernel(grayscale_image6, kernel_horizontal)
edges_vertical = apply_kernel(grayscale_image6, kernel_vertical)

edges_horizontal = cv2.convertScaleAbs(edges_horizontal)
edges_vertical = cv2.convertScaleAbs(edges_vertical)

# Plotting the images
plt.figure(figsize=(12, 4))

plt.subplot(1, 3, 1)
plt.imshow(grayscale_image6, cmap='gray')
plt.title('Original Image')

plt.subplot(1, 3, 2)
plt.imshow(edges_horizontal, cmap='gray')
plt.title('Horizontal Edges')

plt.subplot(1, 3, 3)
```

```
plt.imshow(edges_vertical, cmap='gray')
plt.title('Vertical Edges')

plt.show()
```



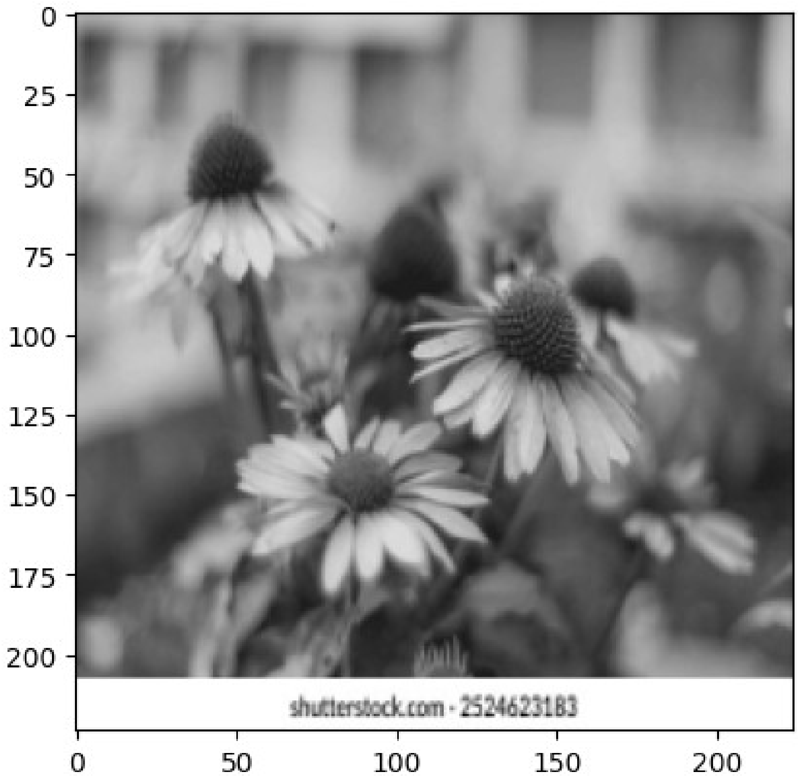
###Explanation: ___

- Here, the Horizontal Kernel only filters the Horizontal Edges.
- The Vertical Kernel only filters the Vertical Edges.
- It's noticable that the increased weights in the middle part (2 and -2) make the edges more visible.

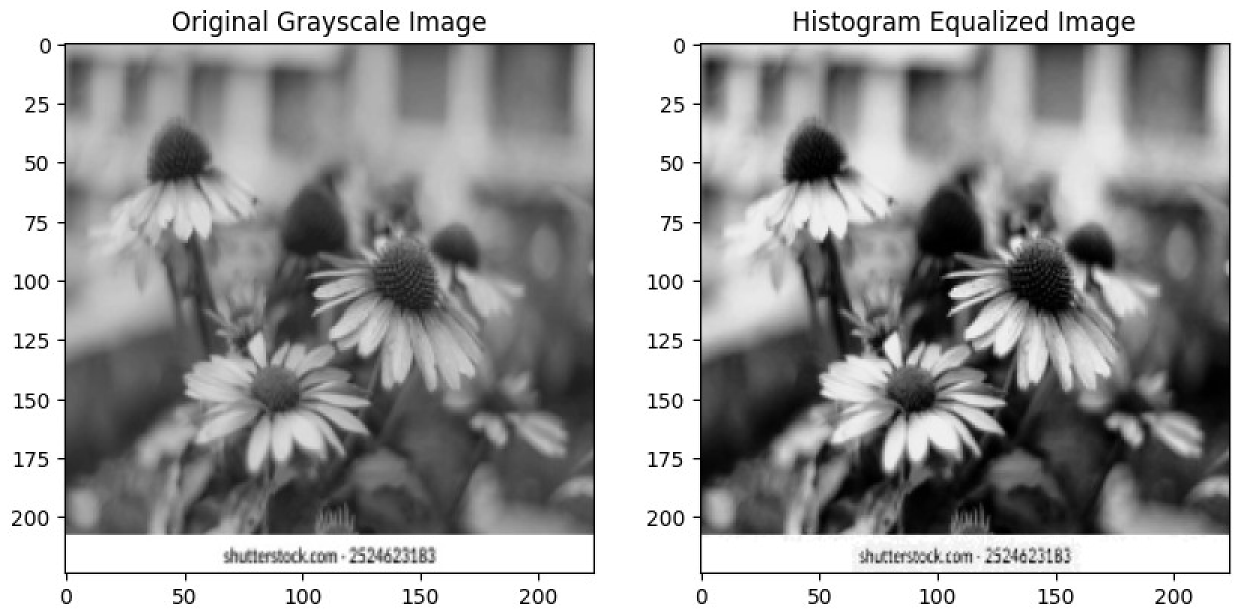
#Task 8

##Sub-task 1

```
image7 = cv2.imread("/content/drive/MyDrive/storage extension/Colab
Notebooks/CSE463/Lab 2/23341134_UdoySaha_Lab2/Images/Image7.jpg")
image7 = cv2.resize(image7, (224, 224), interpolation=cv2.INTER_AREA)
grayscale_image7 = cv2.cvtColor(image7, cv2.COLOR_BGR2GRAY)
plt.imshow(grayscale_image7, cmap='gray')
plt.show()
```




```
# Apply Histogram Equalization
equalized_image7_1 = cv2.equalizeHist(grayscale_image7)


# Plotting the images
plt.figure(figsize=(10, 5))

plt.subplot(1, 2, 1)
plt.imshow(grayscale_image7, cmap='gray')
plt.title('Original Grayscale Image')

plt.subplot(1, 2, 2)
plt.imshow(equalized_image7_1, cmap='gray')
plt.title('Histogram Equalized Image')

plt.show()
```



###Explanation: ___

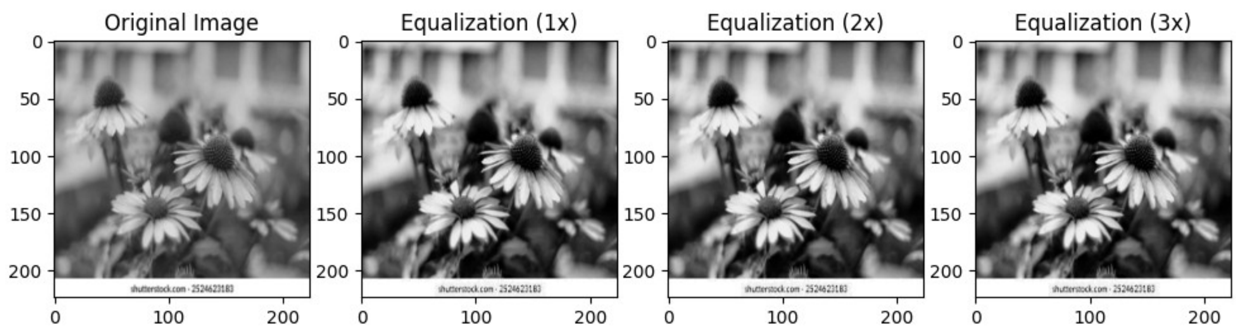
- Contrast basically means the difference of pixel values. The higher the difference is, the higher the contrast is.

- Here, the dark pixels get darker and light pixels get lighter. It tries to distribute the available pixels throughout the whole range [0, 255] and makes the histogram as straight as possible.

##Sub-task 2

```
equalized_image7_2 = cv2.equalizeHist(equalized_image7_1)
equalized_image7_3 = cv2.equalizeHist(equalized_image7_2)

# Plotting the images
plt.figure(figsize=(12, 4))

plt.subplot(1, 4, 1)
plt.imshow(grayscale_image7, cmap='gray')
plt.title('Original Image')

plt.subplot(1, 4, 2)
plt.imshow(equalized_image7_1, cmap='gray')
plt.title('Equalization (1x)')

plt.subplot(1, 4, 3)
plt.imshow(equalized_image7_2, cmap='gray')
```

```
plt.title('Equalization (2x)')

plt.subplot(1, 4, 4)
plt.imshow(equalized_image7_3, cmap='gray')
plt.title('Equalization (3x)')

plt.show()
```



###Explanation: ___

- After the first equalization, there are no drastic changes in the later equalizations. Because, the histogram is already tried to make as straight as possible in the first equalizing step.
- The 3rd equalization shows a bit white and black bands due to over equalization.

- There is no visible details appeared after the first equalization.