Aim: To study and perform Morphological operations on an image

1. Erosion
2. Dilation
3. Opening
4. Closing

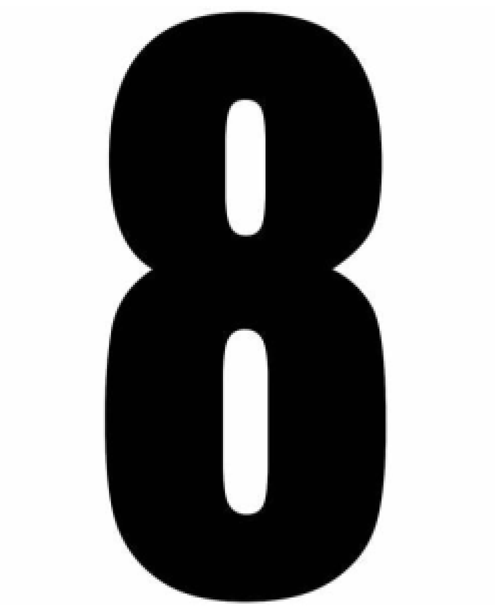
In [ ]:
```
import cv2
import numpy as np
from google.colab.patches import cv2_imshow
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call driv
e.mount("/content/drive", force_remount=True).
```

Dilation

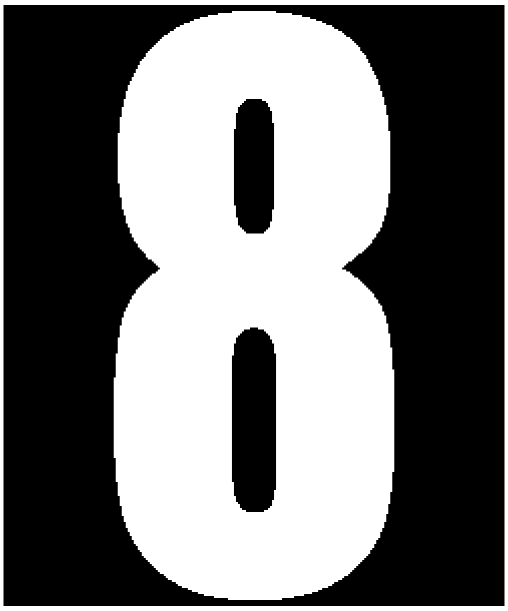
In [ ]:
```
org = cv2.imread('/content/drive/MyDrive/Colab Notebooks/8_binary_img.jpg',0)
img = cv2.resize(org, (250,300), interpolation = cv2.INTER_AREA)
print('Binary Image\n')
cv2_imshow(img)
```

```
Binary Image
```

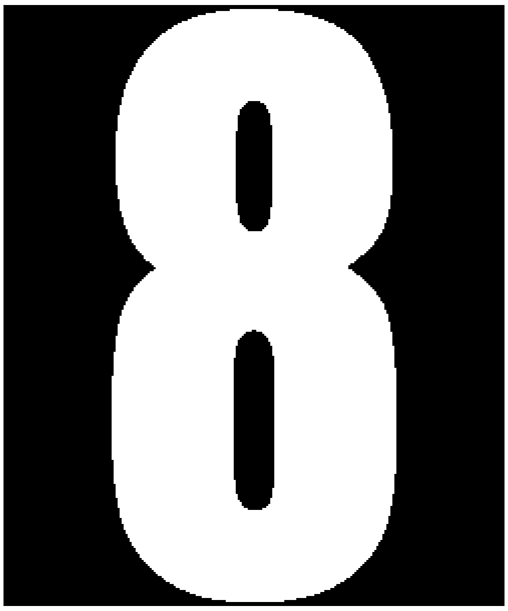


```
In [ ]: # Binarize the image
        binr = cv2.threshold(img,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]
        # Define the kernel 3x3
        kernel = np.ones((3,3),np.uint8)
        # invert the image
        invert = cv2.bitwise_not(binr)
        print('Inverted Binary Image\n')
        cv2_imshow(invert)
        # dilate the image
        dilation = cv2.dilate(invert, kernel, iterations=1)
        print('\nDilated Binary Image\n')
        cv2_imshow(dilation)
```

```
Inverted Binary Image
```



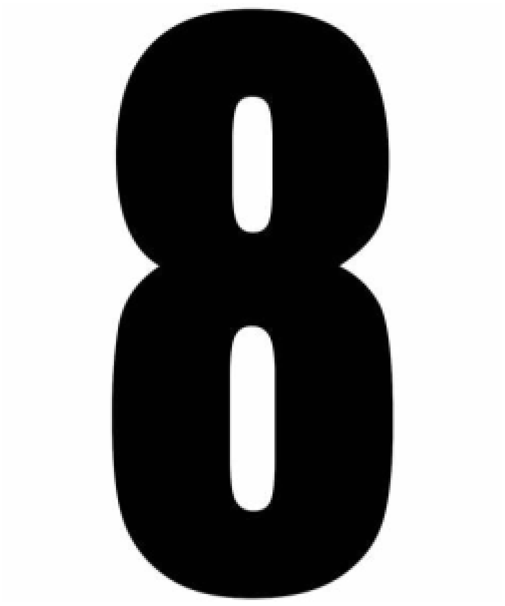
```
Dilated Binary Image
```



Erosion

```
In [ ]: print('Binary Image\n')
        cv2_imshow(img)
```

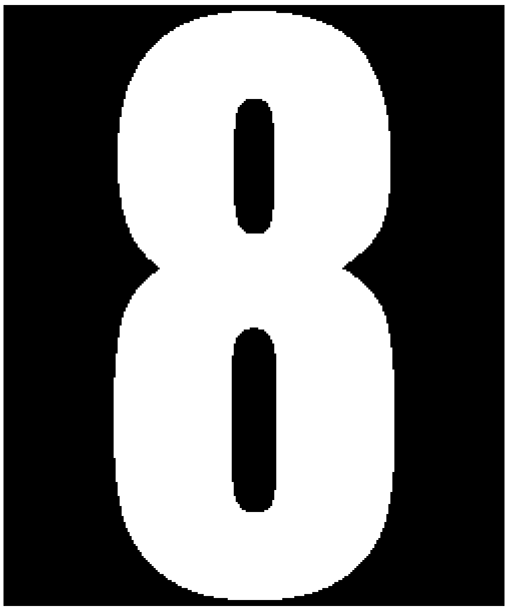
```
Binary Image
```



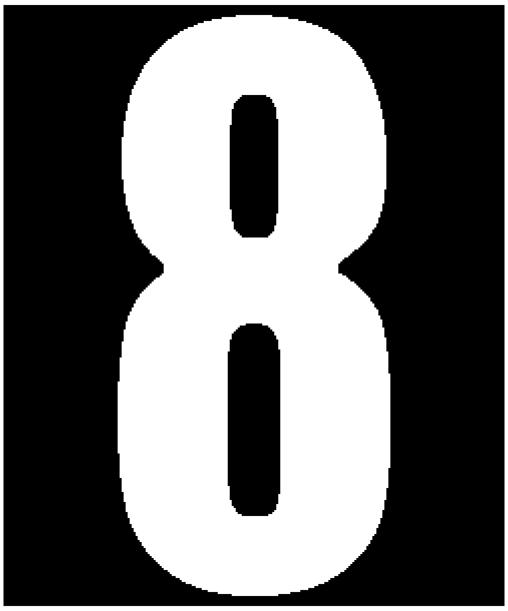
In [ ]:
```python
# Binarize the image
binr = cv2.threshold(img,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]

# Define the kernel 5x5
kernel = np.ones((5,5),np.uint8)
# invert the image
invert = cv2.bitwise_not(binr)
print('Inverted Binary Image\n')
cv2_imshow(invert)
# dilate the image
erosion = cv2.erode(invert, kernel, iterations=1)
print('\nEroded Binary Image\n')
cv2_imshow(erosion)
```

```
Inverted Binary Image
```



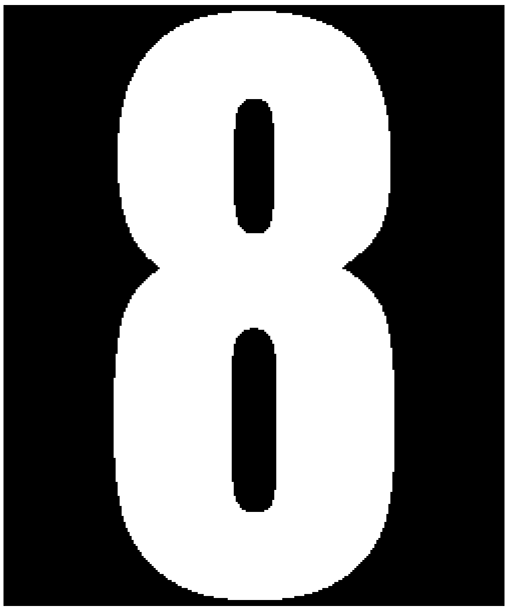
```
Eroded Binary Image
```



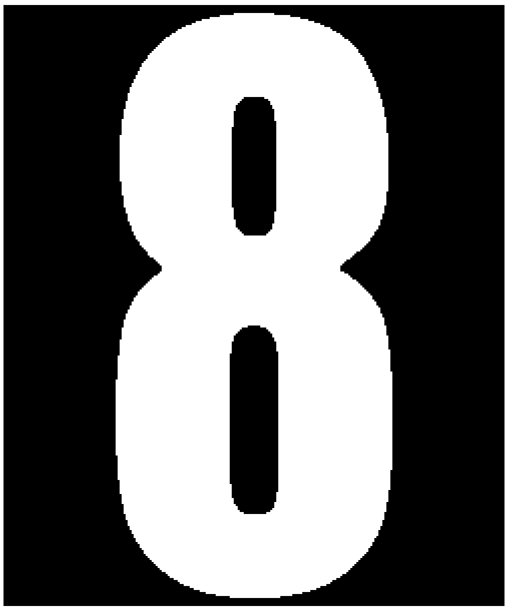
Erosion with 3X3 mask

```
In [ ]: # Binarize the image
        binr = cv2.threshold(img,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]
        # Define the kernel 3x3
        kernel = np.ones((3,3),np.uint8)
        # invert the image
        invert = cv2.bitwise_not(binr)
        print('Inverted Binary Image\n')
        cv2_imshow(invert)
        # dilate the image
        erosion = cv2.erode(invert, kernel, iterations=1)
        print('\nEroded Binary Image with 3X3 mask\n')
        cv2_imshow(erosion)
```

```
Inverted Binary Image
```



```
Eroded Binary Image with 3X3 mask
```

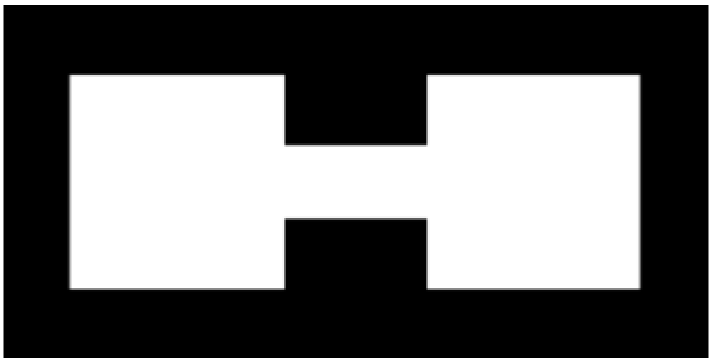


Opening

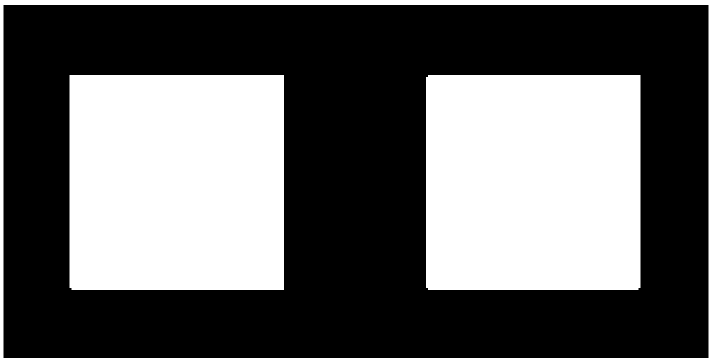
```
In [ ]: imgH = cv2.imread('/content/drive/MyDrive/Colab Notebooks/opening-H.png',0)
        print('Binary Image\n')
        cv2_imshow(imgH)

        # Binarize the image
        binr = cv2.threshold(imgH,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]
        # Define the kernel 3x3
        kernel = np.ones((3,3),np.uint8)
        # opening the image
        opening = cv2.morphologyEx(binr,cv2.MORPH_OPEN, kernel, iterations=25)
        print('\nOpened Binary Image with 3X3 mask\n')
        cv2_imshow(opening)
```

```
Binary Image
```



```
Opened Binary Image with 3X3 mask
```

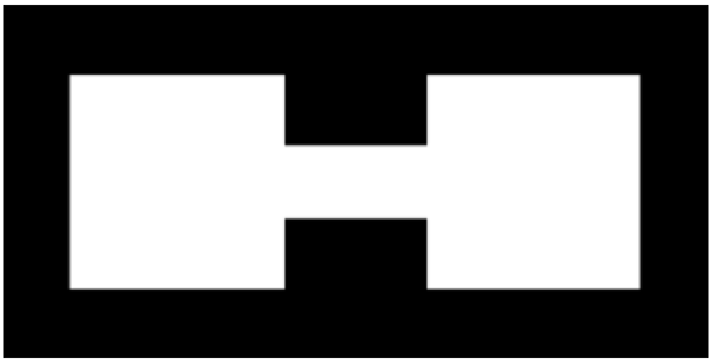


Closing

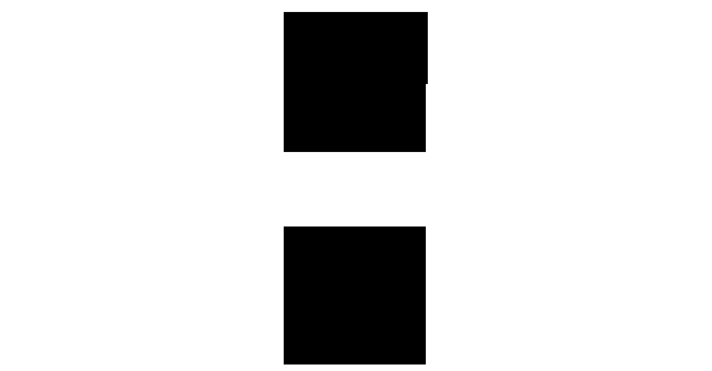
```
In [ ]:
imgH = cv2.imread('/content/drive/MyDrive/Colab Notebooks/opening-H.png',0)
print('Binary Image\n')
cv2_imshow(imgH)

# Binarize the image
binr = cv2.threshold(imgH,0,255,cv2.THRESH_BINARY+cv2.THRESH_OTSU)[1]
# Define the kernel 3x3
kernel = np.ones((3,3),np.uint8)
# closing the image
closing = cv2.morphologyEx(binr,cv2.MORPH_CLOSE, kernel, iterations=35)
print('\nClosed Binary Image with 3X3 mask\n')
cv2_imshow(closing)
```

```
Binary Image
```



```
Closed Binary Image with 3X3 mask
```



```
In [ ]:
```