1. Write a program to read an Image and generate the negative of an Image.

```
from PIL import Image
import numpy as np
from PIL import ImageOps

img = Image.open("Z:\CV_Lab\Images\index.jpg")
img.show()
print(img.mode)
print(img.format)
print(img.size)
img_invert = ImageOps.invert(img)
img_invert.save('Z:\CV_Lab\Images\index_invert.jpg')
img_invert.show()
```

2. Read an image and apply thresholding to the image. (Convert to Grayscale and then to binary).

```
from PIL import Image
import numpy as np
from PIL import ImageOps

img = Image.open("Z:\CV_Lab\Images\index.jpg")
img.show()
print(img.mode)
print(img.format)
print(img.size)
grayscale = img.convert('L')
threshold = 0
img_invert_threshold = img.point( lambda p: 0 if p < threshold else p
) threshold = 255
img_invert_threshold = img.point( lambda p: 255 if p > threshold else p
) img_invert_threshold = img_invert_threshold.convert('1')
img_invert_threshold.save('Z:\CV_Lab\Images\img_invert_threshold.jpg'
) img_invert_threshold.show()
```

3. Capture video from the Camera / static video and store continuous frames.

```
import cv2
import os
video = cv2.VideoCapture("Z:/CV_Lab/Images/sample.mp4")
try:
        if not os.path.exists('data'):
        os.makedirs('data')
except OSError:
```

```
        print ('Error')

currentframe = 0
while(True):
        vid,frameNum = video.read()
        if vid:
        frame_name = 'Video_frame' + str(currentframe) + '.jpg'
        print ('Saving...' + frame_name)
        cv2.imwrite(frame_name, frameNum)
        currentframe += 1
        else:
        break

video.release()
cv2.destroyAllWindows()
```

4. Subtract 2 continuous frames which you got from the previous question.

```
from PIL import Image
import numpy as np
from PIL import ImageOps

img1 = Image.open("Z:/CV_Lab/Video_frame50.jpg")
img1.show()
img2 = Image.open("Z:/CV_Lab/Video_frame51.jpg")
img2.show()
img1_arr = np.asarray(img1)
print(img1_arr)
img2_arr = np.asarray(img2)
print(img2_arr)

frame_diff = img1_arr - img2_arr
sub_image = Image.fromarray(frame_diff)
sub_image.show()
```

5. Image addition

```
from PIL import Image
import numpy as np
from PIL import ImageOps

img1 =
Image.open("Z:\CV_Lab\Images\index.jpg")
img1.show()
img2 =
Image.open("Z:\CV_Lab\Images\index.jpg")
img2.show()
```

```
img1_arr = np.asarray(img1)
print(img1_arr)
img2_arr = np.asarray(img2)
print(img2_arr)

addition = img1_arr + img2_arr
add_image = Image.fromarray(addition)
add_image.save('Z:/CV_Lab/Images/image_addition.jpg')
add_image.show()
```

6. Image Subtraction

```
from PIL import Image
import numpy as np
from PIL import ImageOps

img1 =
Image.open("Z:\CV_Lab\Images\index.jpg")
img1.show()
img2 =
Image.open("Z:\CV_Lab\Images\index.jpg")
img2.show()
img1_arr = np.asarray(img1)
print(img1_arr)
img2_arr = np.asarray(img2)
print(img2_arr)

suntraction = img1_arr - img2_arr
sub_image = Image.fromarray(suntraction)
sub_image.save('Z:/CV_Lab/Images/image_suntraction.jpg')
sub_image.show()
```

7. Image Transformation

a. Translation

```
from PIL import Image
import numpy as np
from PIL import ImageOps

img =
Image.open("Z:\CV_Lab\Images\index.jpg")
img.show()
#translate
out1 = img.point(lambda i: i * 0.5)
out1.save('Z:/CV_Lab/Images/image_translation.jpg')
out1.show()
```

b. Rotation

```
from PIL import Image
import numpy as np
from PIL import ImageOps

img =
Image.open("Z:\CV_Lab\Images\index.jpg")
img.show()
#rotate
out3 = img.rotate(45)
out3.save('Z:/CV_Lab/Images/image_rotation.jpg')
out3.show()
```

c. Resize

```
from PIL import Image
import numpy as np
from PIL import ImageOps


img =
Image.open("Z:\CV_Lab\Images\index.jpg")
img.show()
#resize
out2 = img.resize((128, 128))
out2.save('Z:/CV_Lab/Images/image_resizing.jpg')
out2.show()
```

8. ROI Selection

```
img =
Image.open("Z:/CV_Lab/Images/tiger.jpg")
img.show()

imageName="Z:/CV_Lab/Images/tiger.jpg"
with Image.open(imageName) as my_image:
        cropped_image = my_image.crop((40, 100, 200, 150))
        cropped_image_arr=np.asarray(cropped_image)
        my_image_arr=np.asarray(my_image)
        cropped_image_arr=cropped_image_arr * 5
        my_image_arr[100:150,40:200]=cropped_image_ar
        r out=Image.fromarray(my_image_arr)
        out.show()
```

9. Write your own procedure to apply thresholding to an image.

```
from PIL import Image
import numpy as np
from PIL import ImageOps

img =
Image.open("Z:\CV_Lab\Images\index.jpg")
img.show()
print(img.mode)
print(img.format)
print(img.size)
threshold = 128
grayscale = img.convert('L')
img_arr = np.asarray(grayscale)
img_arr[img_arr>127] = 255
img_arr[img_arr<=127] = 0
image = Image.fromarray(img_arr)
image.show()
```

10. Perform Logical operations like AND, OR, NOT on images.

```
from PIL import Image
from PIL import ImageChops
import numpy as np

img1 = Image.open("Z:\CV_Lab\Images\index.jpg")
img1.show()
img2 = Image.open("Z:\CV_Lab\Images\index.jpg")
img2.show()
image1_array = np.asarray(img1)
image2_array = np.asarray(img1)


#AND
and_arr = np.logical_and(image1_array, image2_array).astype(np.uint8)
and_image = Image.fromarray(and_arr)
and_image.show()
#OR
or_arr = np.logical_or(image1_array,
image2_array).astype(np.uint8) or_image = Image.fromarray(or_arr)
or_image.show()
#NOT
not_array =
np.logical_not(image1_array).astype(np.uint8) not_image
= Image.fromarray(not_array)
not_image.show()
```