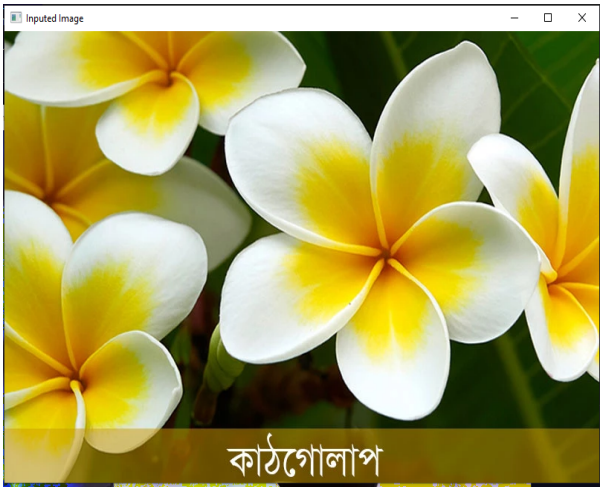
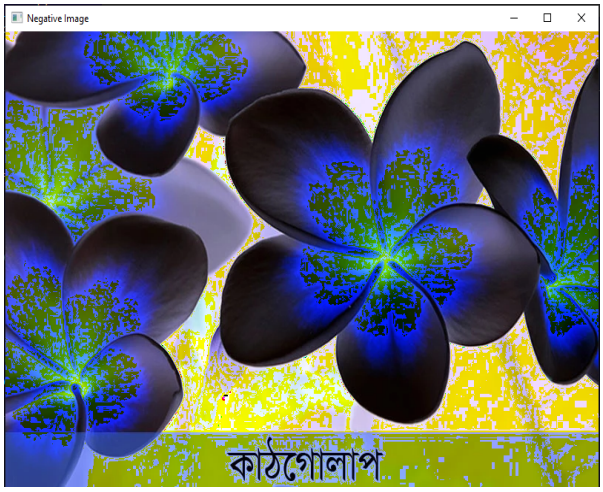
LAB REPORT

Image Processing and Computer Vision Sessional

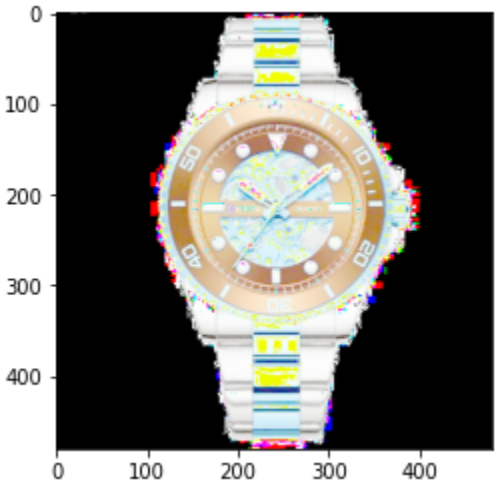
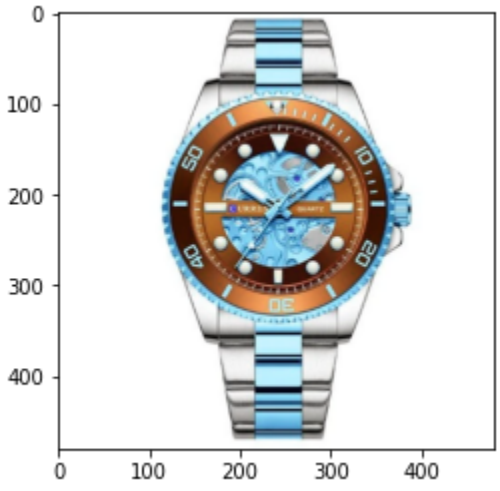
Date: 15-12-2022

Submitted To	Submitted by
Md. Rashedul Islam Lecturer, Dept. CSE Sonagaon University	MD Shahin Mia Robin Section: 16A2 ID: CSE1901016113

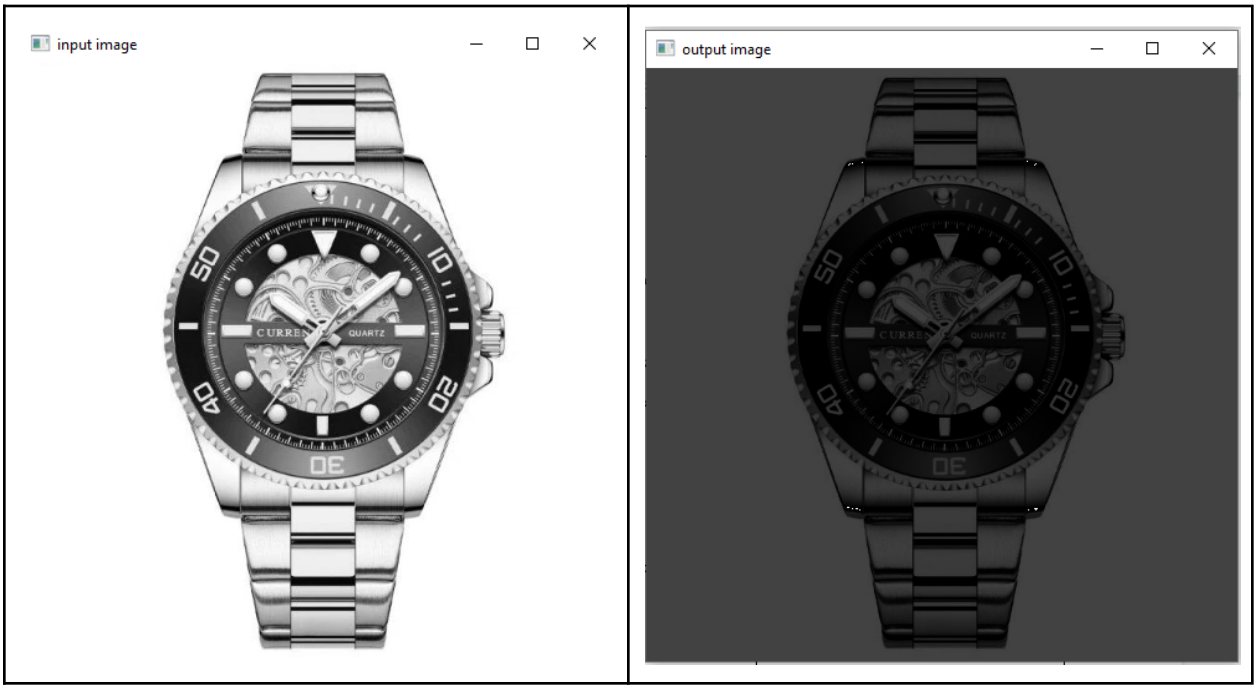
```
import cv2
import math

inputImage = cv2.imread('../images/kath_golap2.jpg', 1)

cv2.imshow('Inputed Image', inputImage)

# subtracting the original image

img_neg = 1 - inputImage

cv2.imshow('Negative Image', img_neg)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input Image	Output Image
	

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

# Read an image
image = cv2.imread('../images/currenmenswatch1.png')

# Apply log transformation method
c = 255 / np.log(1 + np.max(image))
log_image = c * (np.log(image + 1))

log_image = np.array(log_image, dtype = np.uint8)

# Display both images
plt.imshow(image)
plt.show()
plt.imshow(log_image)
plt.show()
```

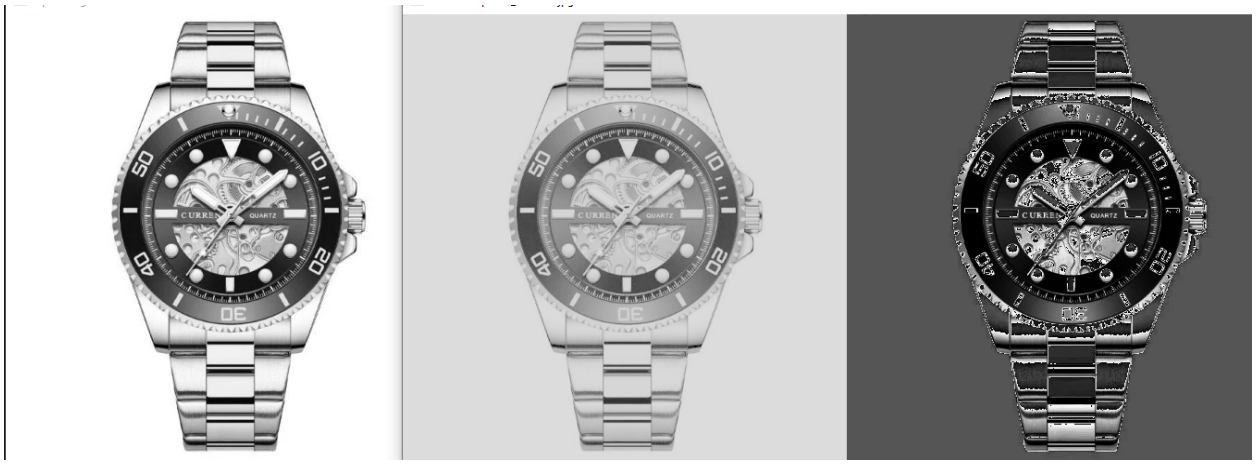
Input Image	Output Image
	

```
import cv2
import math

# Read an image
inputImage = cv2.imread('../images/currenmenswatch1.png', 0)
out = inputImage.copy()
c = 31
cv2.imshow('input image', inputImage)

for xAxis in range(inputImage.shape[0]):
    for yAxis in range(inputImage.shape[1]):
        eachPixel = inputImage.item(xAxis, yAxis)

        d = math.pow(eachPixel / c, 2)
        r = d - 1
        out.itemset((xAxis, yAxis), r)

cv2.imshow('output image', out)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

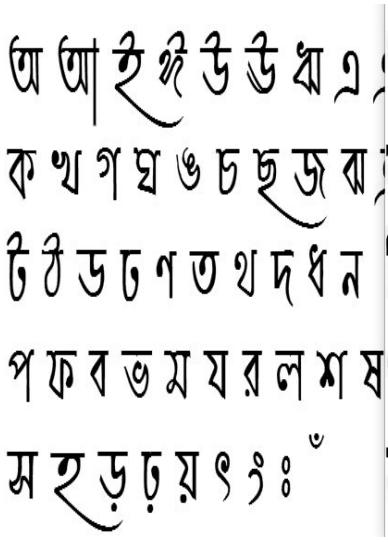
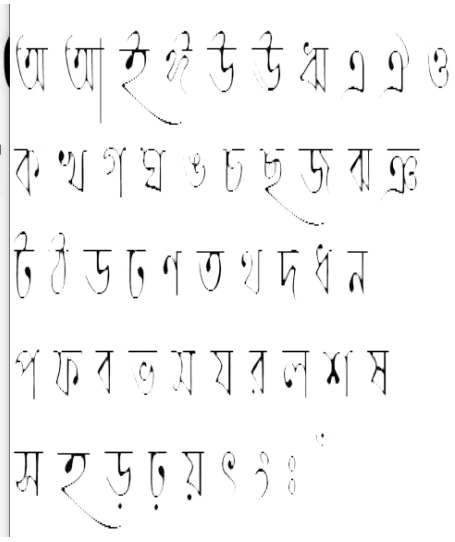


```
import cv2
import math
img      =      cv2.imread('../images/currenmenswatch1.png',
cv2.IMREAD_GRAYSCALE)
brightImage = img.copy()
darkImage = img.copy()

c = 200 # positive constant

cv2.imshow('input image',img)

for xAxis in range(img.shape[0]):
    for yAxis in range(img.shape[1]):
        eachPixel = img.item(xAxis, yAxis) # S

        gamma_one = (c*(eachPixel/c) ** 0.4)
        gamma_two = (c*(eachPixel/c) ** 2.2)

        brightImage.itemset((xAxis, yAxis), gamma_one)
        darkImage.itemset((xAxis, yAxis), gamma_two)

im_h = cv2.hconcat([brightImage, darkImage])
cv2.imshow('data/dst/opencv_hconcat.jpg', im_h)
```

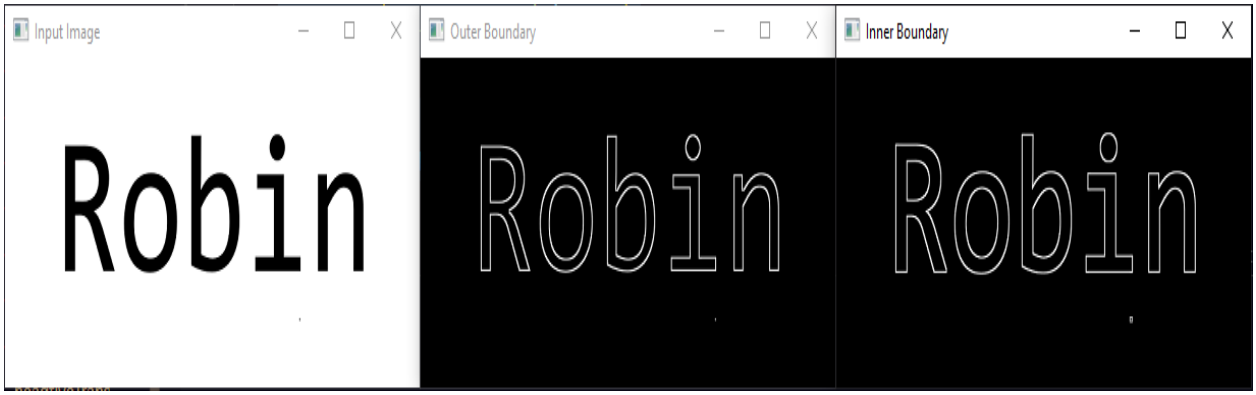
Input Image	Bright Image	Dark Image



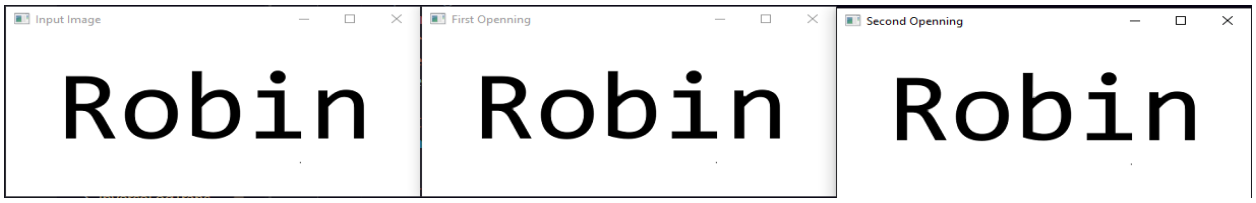
```
import cv2
import numpy

inputImg = cv2.imread('../images/oakkha.jpg', 0)
cv2.imshow('Input Image', inputImg)

kernel1 = numpy.ones((5, 5), numpy.uint8)
kernel2 = numpy.ones((3, 3), numpy.uint8)

inputImg_erosion = cv2.erode(inputImg, kernel1, iterations =
1)
inputImg_dilation = cv2.dilate(inputImg, kernel2, iterations
= 1)

cv2.imshow('After Erosion', inputImg_erosion)
cv2.imshow('After Dilation', inputImg_dilation)

cv2.waitKey(0)
cv2.destroyAllWindows()
```

Input Image	After Erosion	After Dilation
		

```
import cv2
import numpy

inputImg          =          cv2.imread('../images/name.png',
cv2.IMREAD_GRAYSCALE)
cv2.imshow('Input Image', inputImg)

kernel1 = numpy.ones((3, 3), numpy.uint8)

# inner boundary
inputImg_erosion=cv2.erode(inputImg, kernel1, iterations=1)
inner = inputImg - inputImg_erosion

# outer boundary
inputImg_dilation = cv2.dilate(inputImg, kernel1, iterations
= 1)
outer = inputImg_dilation - inputImg

cv2.imshow('Inner Boundary', inner)
cv2.imshow('Outer Boundary', outer)

cv2.waitKey(0)
cv2.destroyAllWindows()
```



```
import cv2
import numpy
# opening custom function
def imageOpenning(image, karnel, loop):
    erosion = cv2.erode(image, karnel, loop)
    result = cv2.dilate(erosion, karnel, loop)
    return result
inputImg = cv2.imread('../images/name.png', cv2.IMREAD_GRAYSCALE)
cv2.imshow('Input Image', inputImg)
kernel = numpy.ones((3, 3), numpy.uint8)

firstOpenning = imageOpenning(inputImg, kernel, 1)
secondOpenning = imageOpenning(firstOpenning, kernel, 1)

cv2.imshow('First Openning', firstOpenning)
cv2.imshow('Second Openning', secondOpenning)
```




```
# Closing custom function
def imageClosing(image, karnel, loop):
    dialation = cv2.dilate(image, karnel, loop)
    result = cv2.erode(dialation, karnel, loop)
    return result
inputImg = cv2.imread('../images/blur_image.jpg',
cv2.IMREAD_UNCHANGED)
cv2.imshow('Input Image', inputImg)
kernel = numpy.ones((3, 3), numpy.uint8)

firstClosing = imageClosing(inputImg, kernel, 1)
secondClosing = imageClosing(firstClosing, kernel, 1)

cv2.imshow('First Closing', firstClosing)
cv2.imshow('Second Closing', secondClosing)
```