**DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING**

**COLLEGE OF E&ME, NUST, RAWALPINDI**

**Subject Name**

**Digital Image Processing**

**Lab Number**

**3**

**SUBMITTED TO:**

**Dr. Asad Khan**

**LE Sundas Ashraf**

**SUBMITTED BY:**

**Student Name**

1. Wahaaj Nasir

**Reg#413238**

**DE- 44 Dept C&SE**

**Objectives:**

Basics of Image Processing in Python

**Related Topic/Chapter in theory class:**

Basics Of Digital Image Processing

**Hardware/Software required:**

Hardware: PC

Software Tool: Pycharm

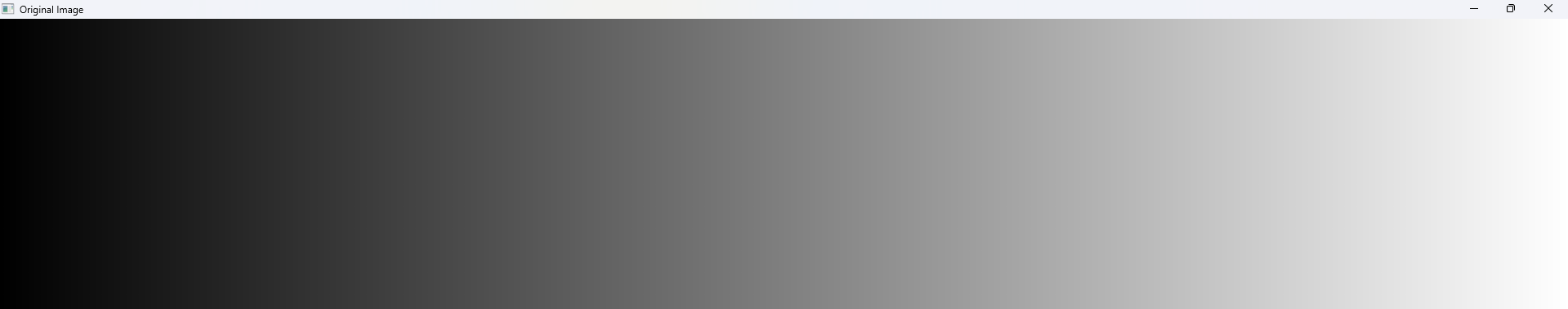
**Task 1:**

**Read an image and convert it into black & white (binary image). Calculate threshold value by taking mean of whole image. Also, convert the same image into a negative image using the transformation: s = (L − 1) – r where L are the total grayscale levels of the image. Use the gradient image that we used in the last lab.**

**Solution:**

import numpy as np  
import cv2 as cv  
  
def lower\_by\_2(image):  
 rows, cols = image.shape  
 new\_image = np.ones((rows, cols), dtype=np.uint8)  
  
 for i in range(rows):  
 for j in range(cols):  
 if (image[i, j] >= 0 and image[i, j] <= 127):  
 new\_image[i, j] = 0  
 elif (image[i, j] >= 128 and image[i, j] <= 255):  
 new\_image[i, j] = 255  
  
 return new\_image  
  
*# s = (L-1)-r*def neg\_img(image):  
 l = 256  
 rows, cols = image.shape  
 new\_img = np.zeros((rows, cols), dtype = np.uint8)  
 for i in range(rows):  
 for j in range(cols):  
 r = int(image[i][j])  
 s = (256-1)-r  
 new\_img[i][j] = np.uint8(s)  
  
 return new\_img  
  
*#Main*image = cv.imread("D:/Uni/Semester 6/DIP/Self/Lab/Lab 3/Lab 3/gradient.png", 0)  
image\_bin = lower\_by\_2(image)  
  
thresh = np.mean(image)  
thresh\_bin = np.mean(image\_bin)  
print(f"Threshold value: {thresh}")  
print(f"Threshold value of bin img: {thresh\_bin}")  
  
image\_neg = neg\_img(image)  
  
cv.imshow('Original Image', image)  
cv.imshow('Negative Image', image\_neg)  
cv.imshow('2 Levels', image\_bin)  
cv.waitKey()

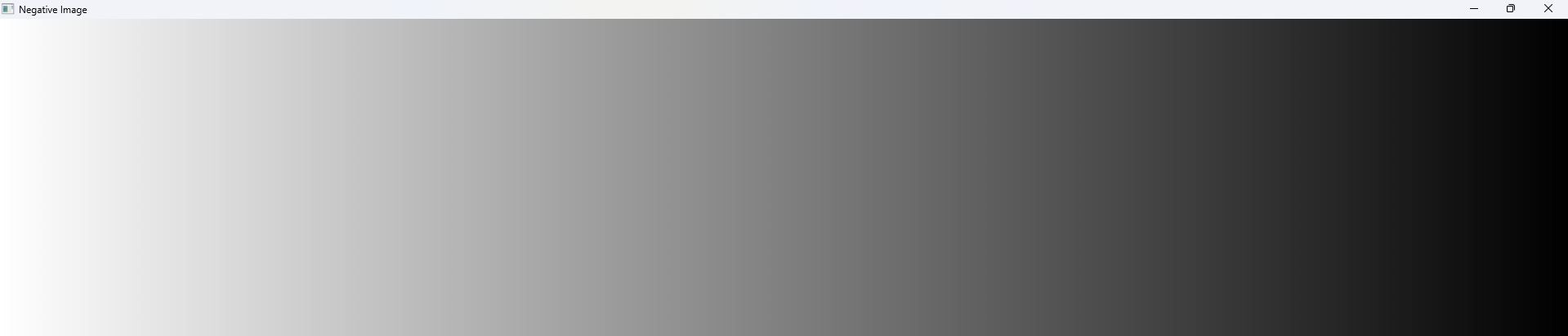
**Output:**

****

**Original**

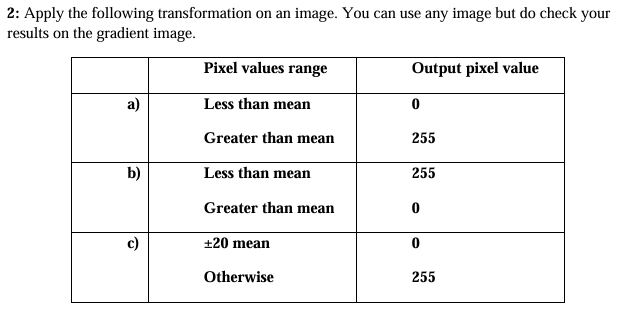
****

**2 Levels (Binary)**

****

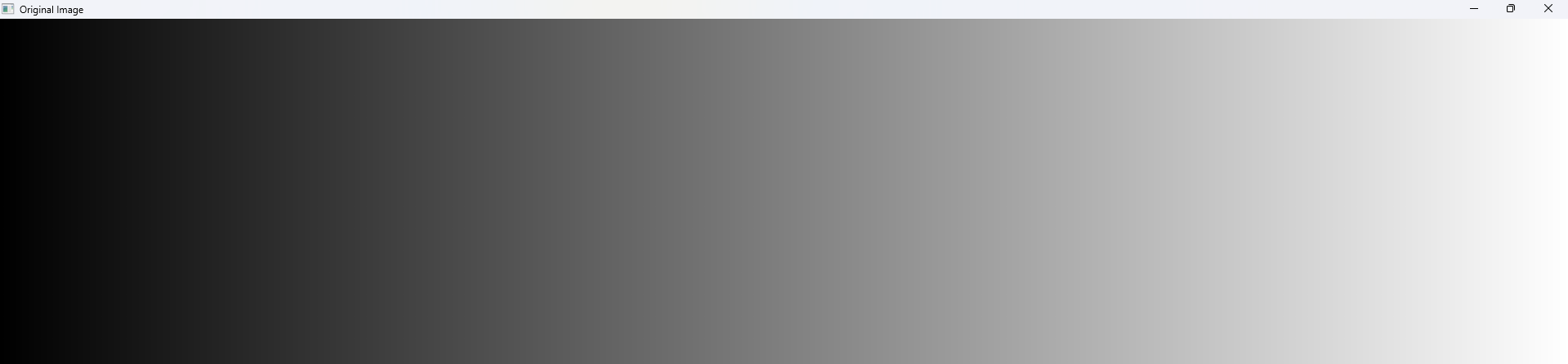
**Neg Image**

**Task 2:**

**Solution**

import numpy as np  
import cv2 as cv  
  
def img\_mean\_transforms(image):  
 rows, cols = image.shape  
 mean = int(np.mean(image))  
 print(mean)  
 img1 = np.zeros((rows, cols), dtype = np.uint8)  
 img2 = np.zeros((rows, cols), dtype=np.uint8)  
 img3 = np.zeros((rows, cols), dtype=np.uint8)  
  
 *#Cond 1* for i in range(rows):  
 for j in range(cols):  
 val = int(image[i][j])  
 if((val >= 0) & (val <= mean)):  
 img1[i][j] = 0  
 elif((val > mean) & (val <= 255)):  
 img1[i][j] = 255  
  
 *#Cond 2* for i in range(rows):  
 for j in range(cols):  
 val = int(image[i][j])  
 if ((val >= 0) & (val <= mean)):  
 img2[i][j] = 255  
 elif ((val > mean) & (val <= 255)):  
 img2[i][j] = 0  
  
 *#Cond 3* for i in range(rows):  
 for j in range(cols):  
 val = int(image[i][j])  
 if((val >= mean-20) & (val <= mean+20)):  
 img3[i][j] = 0  
 else:  
 img3[i][j] = 255  
  
 cv.imshow('Original Image', image)  
 cv.imshow('Cond 1', img1)  
 cv.imshow('Cond 2', img2)  
 cv.imshow('Cond 3', img3)  
 cv.waitKey()  
  
*#Main*image = cv.imread("D:/Uni/Semester 6/DIP/Self/Lab/Lab 3/Lab 3/gradient.png", 0)  
img\_mean\_transforms(image)

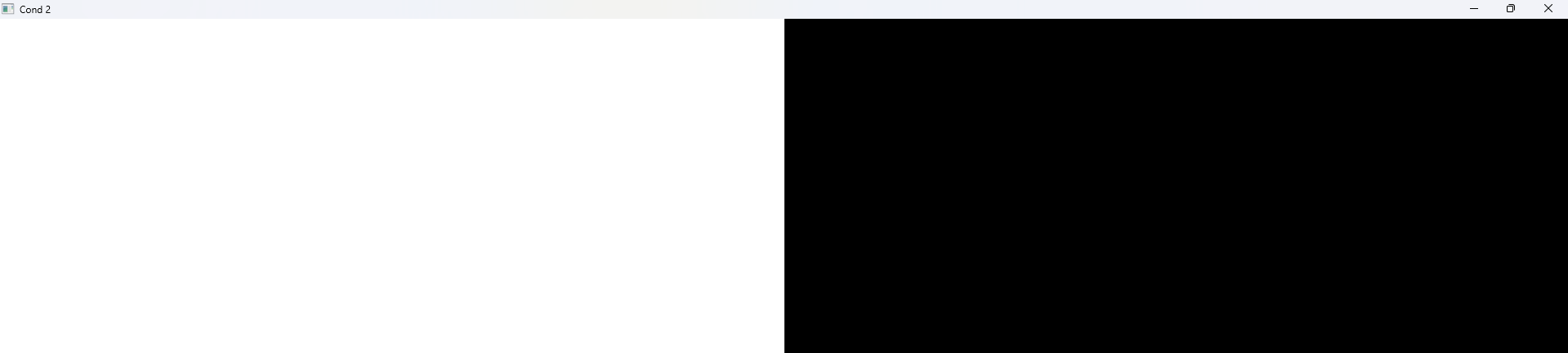
**Output:**



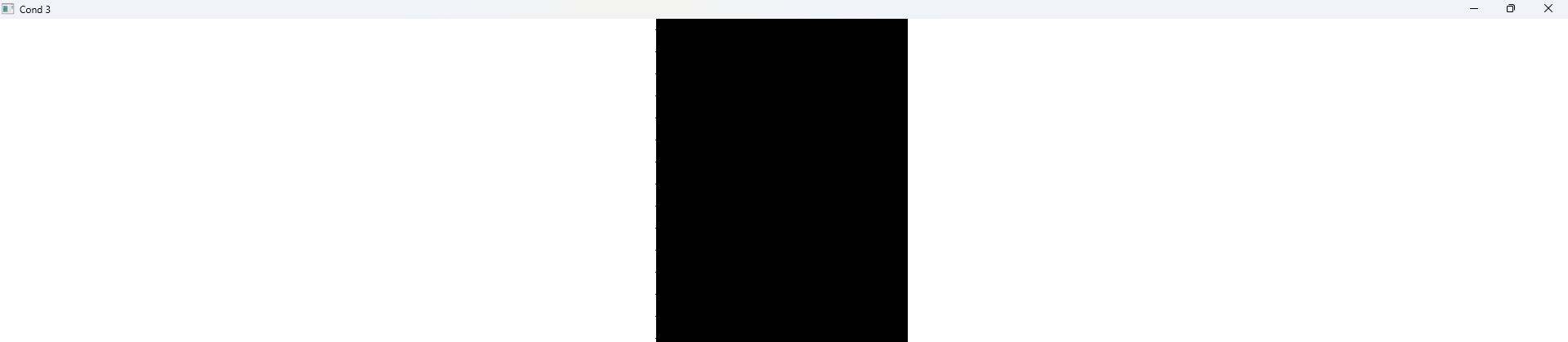
**Original Image**

****

**Condition 1**

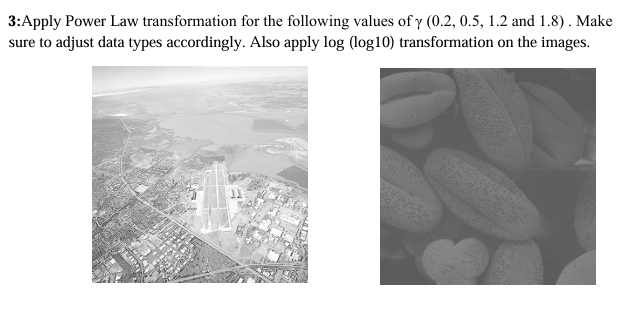
****

**Condition 2**

****

**Condition 3**

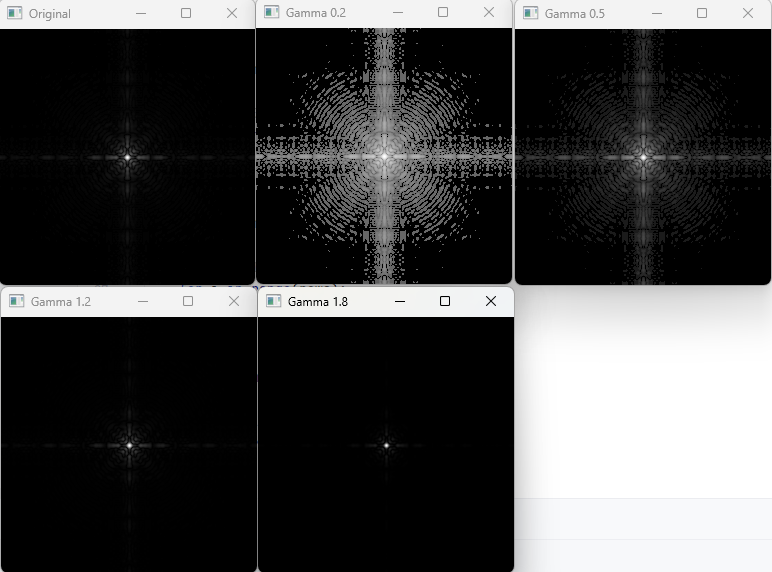
**Task 3:**



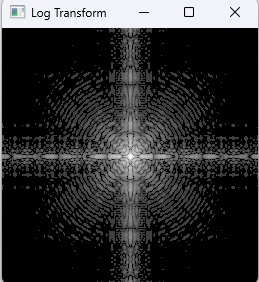
**Solution**

import numpy as np  
import cv2 as cv  
import math  
  
def transform\_power(image):  
 rows, cols = image.shape  
 img1 = np.zeros((rows, cols), dtype=np.uint8)  
 img2 = np.zeros((rows, cols), dtype=np.uint8)  
 img3 = np.zeros((rows, cols), dtype=np.uint8)  
 img4 = np.zeros((rows, cols), dtype=np.uint8)  
  
 *#Gamma = 0.2* for i in range(rows):  
 for j in range(cols):  
 r = image[i][j]  
 s = int(255 \* ((r/255)\*\*0.2))  
 img1[i][j] = s  
  
 *#Gamma = 0.5* for i in range(rows):  
 for j in range(cols):  
 r = image[i][j]  
 s = int(255 \* ((r/255)\*\*0.5))  
 img2[i][j] = s  
  
 *#Gamma = 1.2* for i in range(rows):  
 for j in range(cols):  
 r = image[i][j]  
 s = int(255 \* ((r/255)\*\*1.2))  
 img3[i][j] = s  
  
 *#Gamma = 1.8* for i in range(rows):  
 for j in range(cols):  
 r = image[i][j]  
 s = int(255 \* ((r/255)\*\*1.8))  
 img4[i][j] = s  
  
 cv.imshow('Original', image)  
 cv.imshow('Gamma 0.2', img1)  
 cv.imshow('Gamma 0.5', img2)  
 cv.imshow('Gamma 1.2', img3)  
 cv.imshow('Gamma 1.8', img4)  
 cv.waitKey()  
  
def log\_transform(image):  
 c = 255 /np.log10(1+int(np.max(image))) *#1+255 in np.max returns 0 as it wraps around* rows, cols = image.shape  
 new\_img = np.zeros((rows, cols), dtype=np.uint8)  
  
 for i in range(rows):  
 for j in range(cols):  
 r = int(image[i][j])  
 s = c \* (np.log10(r+1))  
 new\_img[i][j] = np.uint8(s)  
  
 return new\_img  
  
*#Main*image = cv.imread("D:/Uni/Semester 6/DIP/Self/Lab/Lab 3/Lab 3/fig03.tif", 0)  
transform\_power(image)  
log\_img = log\_transform(image)  
  
print(np.max(image))  
cv.imshow('Log Transform', log\_img)  
cv.waitKey()

**Output:**

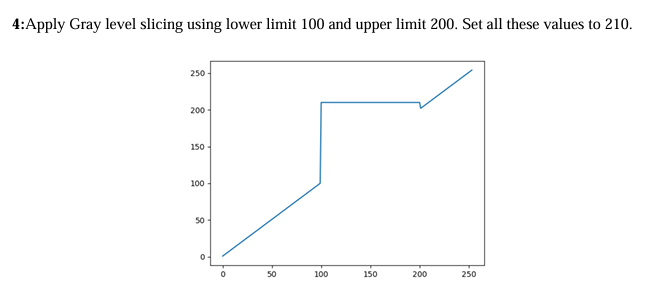
****

**Gamma Results**

****

**Log Transform**

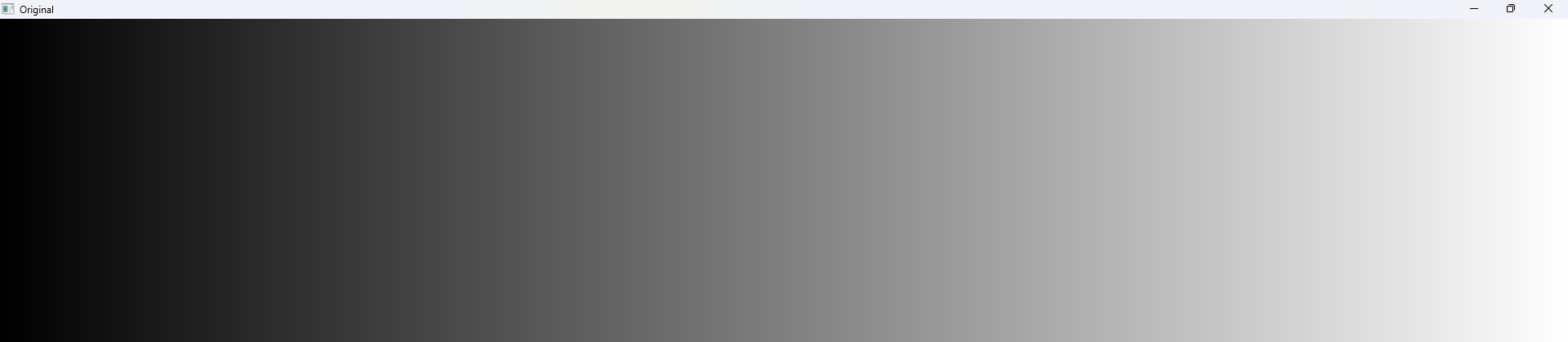
**Task 4:**



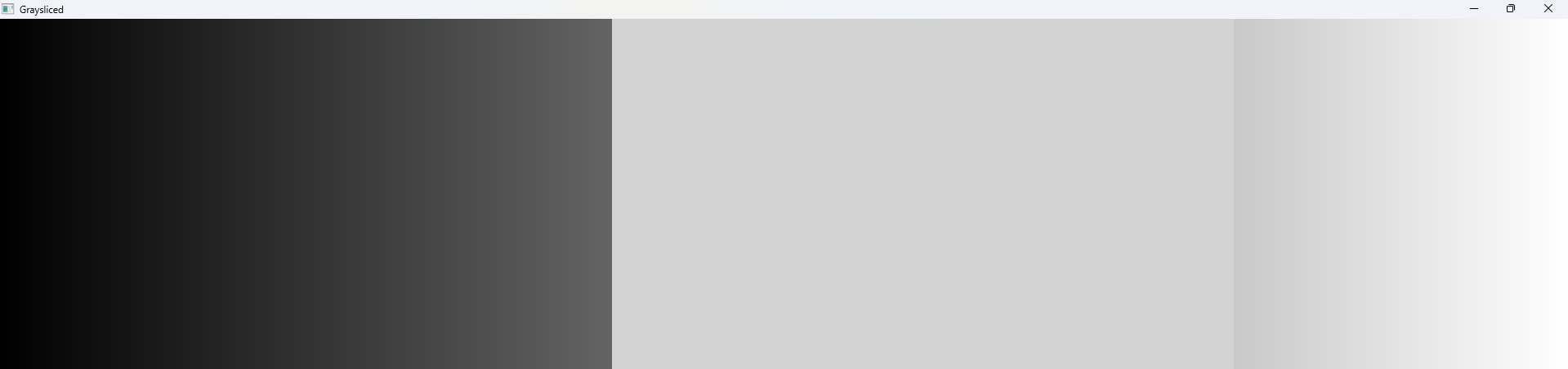
**Solution**

import numpy as np  
import cv2 as cv  
  
def grayslicing(image):  
 rows, cols = image.shape  
 new\_img = np.zeros((rows, cols), dtype=np.uint8)  
 for i in range(rows):  
 for j in range(cols):  
 val = int(image[i][j])  
 if((val >= 100) & (val <= 200)):  
 new\_img[i][j] = 210  
 else:  
 new\_img[i][j] = image[i][j]  
  
 return new\_img  
  
*#Main*image = cv.imread("D:/Uni/Semester 6/DIP/Self/Lab/Lab 3/Lab 3/gradient.png", 0)  
gray\_img = grayslicing(image)  
  
cv.imshow('Original', image)  
cv.imshow('Graysliced', gray\_img)  
cv.waitKey()

**Output:**

****

**Original Image**

****

**Graysliced**