Assignment 4 numpy opencv

Question 1 - 4

```
# 1. Write a program to load a .csv file as a NumPy 1-D array. Find the maximum and minimum elements in the array. Hint: For the data, use the .csv file “book1.csv”
import numpy as np
# read a csv file into numpy 1-D array using genfromtxt
# skip_header=1 to skip the first row of the csv file
# usecols=1 to read the second column of the csv file
# dtype=int to read the elements as integers
data = np.genfromtxt('book1.csv', delimiter='\t', skip_header=1, usecols=1, dtype=int)
# find the maximum and minimum elements in the array
max_element = np.max(data)
min_element = np.min(data)
print('Maximum element of book1.csv:', max_element)
print('Minimum element of book1.csv:', min_element)

#2. For the Numpy 1-D array as obtained in Q.1, sort the elements in ascending order.
# sort the elements in ascending order
sorted_data = np.sort(data)

#3. For the sorted Numpy 1-D array as obtained in Q.2, reverse the array and print the reversed array.
# reverse the array and print the reversed array
# using slicing to reverse the array
reversed_data = sorted_data[::-1]
print('Reversed array of book1.csv:', reversed_data)

#4. Write a program to load three .csv files (Book1.csv, Book2.csv, and Book3.csv) as a list of Numpy 1-D arrays. Print the means of all arrays as a list.
```

```
# read three csv files into a list of numpy 1-D arrays using genfromtxt
data1 = np.genfromtxt('book1.csv', delimiter='\t', skip_header=1,
usecols=1, dtype=float)
data2 = np.genfromtxt('book2.csv', delimiter='\t', skip_header=1,
usecols=1, dtype=float)
data3 = np.genfromtxt('book3.csv', delimiter='\t', skip_header=1,
usecols=1, dtype=float)
data_list = [data1, data2, data3]

# using for in loop to find the mean of all arrays and print it.
mean_list = []
for data in data_list:
    mean = np.mean(data)
    mean_list.append(mean)
print('Mean of all arrays in order:', mean_list)
```

Question 5 - 12

```
import cv2
import numpy as np
#5. Write a program to read an image, store the image in NumPy 2-D
array. For the image consider a.PNG. Display the image. Let the array
be X. Hint: Use OpenCV to work with images.
# read an image using OpenCV and store the image in NumPy 2-D array
X = np.array(cv2.imread('a.png',cv2.IMREAD_COLOR))
# set the window to autosize
# cv2.namedWindow('OUTPUT', cv2.WINDOW_AUTOSIZE)
# display the image

cv2.imshow('Given Image', X)
cv2.waitKey(0)
cv2.destroyAllWindows()

#6. Write a program to convert a color image (say a.PNG) into a
grescale image. Let the greysacle image stored in the Numpy 2-D array
be X. Display the grayscale iamge on the screen. Hint: Greyscale value
of a pixel is the mean of three RGB values of that pixel.
# convert the color image to grayscale
```

```
gray_X = cv2.cvtColor(X, cv2.COLOR_BGR2GRAY)
cv2.imshow('Grey Scale Image', gray_X)
cv2.waitKey(0)
cv2.destroyAllWindows()

# find the width and height of the image a.png and print the width and
height.

# print('Width:', gray_X.shape[1])
# print('Height:', gray_X.shape[0])

#7. Let Y be the transpose matrix of X. Write a program to obtain Z =
X×Y.
# obtain Y = transpose matrix of X
Y = np.transpose(gray_X)
# obtain Z = X×Y
Z = np.matmul(gray_X, Y)
# print the resultant matrix Z
# print(Z)

#8. For all the problems 7, repeat without using NumPy and compare the
computation times doing the same with NumPy.
#using numpy
import time

start = time.time()
Y = np.transpose(gray_X)
numpyZ = np.matmul(gray_X,Y)
end = time.time()
#takes nearly 1 second
print("Time taken using numpy in seconds: ",end - start)

#without using numpy
start = time.time()
# create copy of the grayscale image with dtype=np.float64
gray_X_Copy = gray_X.copy().astype(np.float64)
# find the transpose matrix of the grayscale image
Y = np.zeros((gray_X.shape[1], gray_X.shape[0]), dtype=np.float64)
for i in range(gray_X.shape[0]):
    for j in range(gray_X.shape[1]):
        Y[j,i] = gray_X_Copy[i,j]
# find the resultant matrix Z , matrix multiplication of X and Y
Z = np.zeros((gray_X.shape[0], gray_X.shape[0]), dtype=np.float64)
```

```
for i in range(gray_X.shape[0]):
    for j in range(gray_X.shape[0]):
        for k in range(gray_X.shape[1]):
            Z[i,j] += gray_X_Copy[i,k] * Y[k,j]
end = time.time()
#takes nearly 450 seconds
print("Time taken without using numpy in seconds: ",end - start)

#9. Plot the pixel intensity histogram using the grayscale image. Hint: Use matplotlib to plot
import matplotlib.pyplot as plt
# plot the pixel intensity histogram using the grayscale image
plt.hist(gray_X.ravel(),256,[0,256])
plt.show()

#10. Create a black rectangle at the position [(40,100) top right, (70, 200) bottom left] in the grayscale image and display the image.
# create a black rectangle at the position [(40,100) top right, (70, 200) bottom left] in the grayscale image
# create a copy of the grayscale image
gray_X_Copy = gray_X.copy()
gray_X_Copy[100:200,40:70] = 0
# display the image
cv2.imshow('Grey Scale with black rectangle', gray_X_Copy)
cv2.waitKey(0)
cv2.destroyAllWindows()

#11. For the grayscale image, display the binarized image with thresholds: [50, 70, 100, 150]. Hint: Binarizing is thresholding each pixel value, i.e., if pixel>threshold, then 1 else 0.
# display the binarized image with thresholds: [50, 70, 100, 150]
# first term in output is the threshold value which is not used in the output
_, binarized_image1 = cv2.threshold(gray_X, 50, 255, cv2.THRESH_BINARY)
_, binarized_image2 = cv2.threshold(gray_X, 70, 255, cv2.THRESH_BINARY)
_, binarized_image3 = cv2.threshold(gray_X, 100, 255,
cv2.THRESH_BINARY)
_, binarized_image4 = cv2.threshold(gray_X, 150, 255,
cv2.THRESH_BINARY)
cv2.imshow('Threshold 50', binarized_image1)
cv2.waitKey(0)
cv2.destroyAllWindows()
cv2.imshow('Threshold 70', binarized_image2)
```

```
cv2.waitKey(0)
cv2.destroyAllWindows()
cv2.imshow('Threshold 100', binarized_image3)
cv2.waitKey(0)
cv2.destroyAllWindows()
cv2.imshow('Threshold 150', binarized_image4)
cv2.waitKey(0)
cv2.destroyAllWindows()

#12. Consider the color image stored in a.png. Create a filter of
[[-1,-1,-1][0,0,0][1,1,1]], and multiply this filter to the image and
output the resultant image.
# create a filter of [[-1,-1,-1][0,0,0][1,1,1]], and multiply this
filter to the image
filter = np.array([[-1,-1,-1],[0,0,0],[1,1,1]])
filtered_image = cv2.filter2D(X, -1, filter)
# output the resultant image
cv2.imshow('Filtered Image', filtered_image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```