Assignment 4
Name: Sharanya Chakraborty
Roll No.: 22CS10088

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`"

```
# Program 1: Loading a .csv file as a NumPy 1-D array, finding maximum and
minimum elements
import numpy as np

# Load the data from 'book1.csv' into a NumPy 1-D array
data = np.loadtxt('book1.csv', delimiter='\t', dtype=int, encoding="utf8",
skiprows=1, usecols=range(1, 2))
# Find and print the maximum and minimum elements in the array
print("Maximum: ", np.max(data))
print("Minimum: ", np.min(data))
# -------------------------------------
```

2. For the Numpy 1-D array as obtained in Q.1, sort the elements in ascending order.

```
# Program 2: Sorting elements in ascending order
import numpy as np

sorted_data = np.sort(data)
print(sorted_data)
# -------------------------------------
```

3. For the sorted Numpy 1-D array as obtained in Q.2, reverse the array and print.

```
# Program 3: Reversing the sorted array
import numpy as np

reverse_array = np.flip(sorted_data)
print(reverse_array)
# -------------------------------------
```

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.

```
# Program 4: Loading three .csv files and printing the means of all arrays as
a list
import numpy as np

data1 = np.loadtxt('book1.csv', delimiter='\t', dtype=int, encoding="utf8",
skiprows=1, usecols=range(1, 2))
data2 = np.loadtxt('book2.csv', delimiter='\t', dtype=float, encoding="utf8",
skiprows=1, usecols=range(1, 2))
data3 = np.loadtxt('book3.csv', delimiter='\t', dtype=int, encoding="utf8",
skiprows=1, usecols=range(1, 2))
```

```
# Calculate means and print as a list
means = [np.mean(data1), np.mean(data2), np.mean(data3)]
print("Means: ", means)
# ------------------------------------
```

5. Write a program to read an image, store the image in NumPy 3-D array. For the image, consider `a.PNG`. Display the image. Let the image stored in the NumPy array be X.
Hint: Use OpenCV to work with image.

```
# Program 5: Reading and displaying an image using OpenCV
import numpy as np
import cv2

# Read and display the image
X = cv2.imread('a.png')
cv2.imshow('flower', X)
print("Image displayed")
cv2.waitKey(0)
cv2.destroyAllWindows() # Press any key to exit
# ------------------------------------
```

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.

```
# Program 6: Converting a color image to grayscale and displaying it
import numpy as np
import cv2

# Convert image to grayscale
img = cv2.imread('a.png')
X = np.mean(img, axis=2, dtype=int).astype(np.uint8)

# Display grayscale image
cv2.imshow('Grayscale Image', X)
cv2.waitKey(0)
cv2.destroyAllWindows()
print("Grayscale Image displayed")
# ------------------------------------
```

7. Let Y be the transpose matrix of X. Write a program to obtain Z = X×Y.

```
# Program 7: Matrix multiplication with NumPy
import time as time
import numpy as np
import cv2

img = cv2.imread('a.png')
```

```
X = np.mean(img, axis=2, dtype=int).astype(np.uint8)

# Transpose X to get Y
Y = X.transpose()

# Perform matrix multiplication X * Y
start = time.time()
Z = np.matmul(X.astype(np.uint16), Y.astype(np.uint16))
print(Z)
print("Time taken for matrix multiplication is:", time.time() - start)
# ------------------------------------
```

8. For the problem in Q. 7, write your program without using NumPy library. Compare the computation times doing the same with NumPy and basic programming in Python.

```
# Program 8: Matrix multiplication without using NumPy
import time as time
import numpy as np
import cv2

img = cv2.imread('a.png')
X = np.mean(img, axis=2, dtype=int).astype(np.uint8)

# Transpose X to get Y
Y = X.transpose()

# Define matrices as lists
x = X.tolist()
y = Y.tolist()

# Initialize result matrix
rows1, cols2 = X.shape[0], X.shape[0]
rows2, cols1 = X.shape[1], X.shape[1]
result = [[0 for _ in range(cols2)] for _ in range(rows1)]

# Perform matrix multiplication
start = time.time()
for i in range(rows1):
    for j in range(cols2):
        for k in range(cols1):
            result[i][j] += x[i][k] * y[k][j]

print(result)
print("The time taken for matrix multiplication without NumPy is:",
time.time() - start)
# ------------------------------------
```

9. Plot the pixel intensity histogram of the grescale image stored in X.
Hint: Use matplotlib to plot the histogram.

```
# Program 9: Plotting pixel intensity histogram
import numpy as np
import cv2
import matplotlib.pyplot as plt

img = cv2.imread('a.png')
X = np.mean(img, axis = 2, dtype = int).astype(np.uint8)
```

```
# Flatten the grayscale image to obtain pixel values
pixel_vals = X.flatten()

# Plot histogram
plt.hist(pixel_vals, bins=range(0, 256), color='green', alpha=0.7)
plt.xlabel('Pixel Intensity')
plt.ylabel('Frequency')
plt.title('Pixel Intensity Histogram')
plt.show()
print("Displaying histogram")
# ------------------------------------
```

10. Create a black rectangle at the position [(40,100) top right, (70, 200) bottom left] in the grayscale image. Display the image.

```
# Program 10: Creating a black rectangle on the grayscale image and
displaying it
import numpy as np
import cv2
import matplotlib.pyplot as plt
import matplotlib.patches as patches

img = cv2.imread('a.png')
X = np.mean(img, axis = 2, dtype = int).astype(np.uint8)

# Display grayscale image with rectangle
plt.imshow(X, cmap='gray')
plt.title('Image with Rectangle')
top_left = (40, 200)
width = 70 - 40
height = 100 - 200

# Defining the rectangle and its position
rectangle = patches.Rectangle(top_left, width, height, edgecolor='black',
facecolor='black')
plt.gca().add_patch(rectangle)
plt.show()
print('Rectangle added to the image')
# ------------------------------------
```

11. Using the grayscale image stored in X, transform it into the binarized image with thresholds: [50, 70, 100, 150]. Let the binarized images are stored in Z50, Z70, Z100, and Z150, respectively.
Hint: Binarizing is thresholding each pixel value, i.e., if pixel>threshold, then 1 else 0.

```
# Program 11: Binarizing grayscale image with different thresholds
import numpy as np
```

```
import cv2
import matplotlib.pyplot as plt

img = cv2.imread('a.png')
X = np.mean(img, axis = 2, dtype = int).astype(np.uint8)

thresholds = [50, 70, 100, 150]
binarized_images = {} # Dictionary to store binarized images with the
mentioned variable names

for threshold in thresholds:
    binarized_image = np.where(X > threshold, 255, 0).astype(np.uint8)
    binarized_images[f'Z{threshold}'] = binarized_image

# Display binarized images side by side
plt.figure(figsize=(8, 6))
for i, threshold in enumerate(thresholds):
    plt.subplot(2, 2, i + 1)
    plt.imshow(binarized_images[f'Z{threshold}'], cmap='binary')
    plt.title(f'Threshold: {threshold}')

plt.tight_layout()
plt.show()
print("Displaying binarized images")
# -------------------------------------
```

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 each pixel value in the image. Display the image after filtering.

```
# Program 12: Filtering color image with a predefined filter
import numpy as np
import cv2
import matplotlib.pyplot as plt

# Read the color image
img = cv2.imread('a.png')

# Define the filter
filter = np.array([[-1, -1, -1],
[0, 0, 0],
[1, 1, 1]])

# Apply the filter to each pixel value in the image
for i in range(img.shape[0]):
    for j in range(img.shape[1]):
        img[i, j] = np.matmul(img[i][j], filter)

# Display the filtered image
plt.figure(figsize=(6, 6))
plt.subplot(1, 1, 1)
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title('Filtered Image')
plt.show()
print("Displaying filtered image")
# -------------------------------------
```