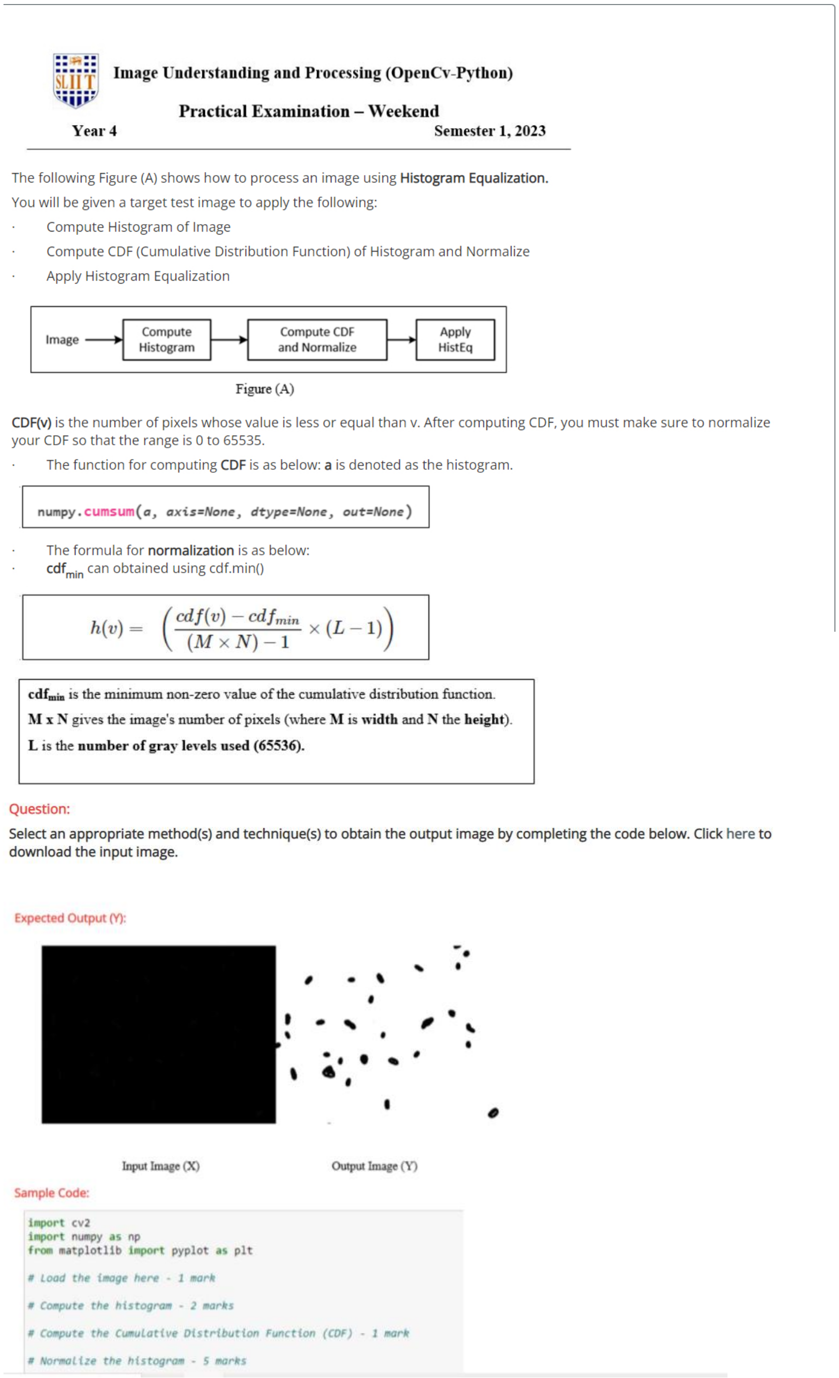

Image Understanding and Processing (OpenCv-Python)

Practical Examination – Weekend

Year 4 **Semester 1, 2023**

The following Figure (A) shows how to process an image using **Histogram Equalization.**

You will be given a target test image to apply the following:

- Compute Histogram of Image
- Compute CDF (Cumulative Distribution Function) of Histogram and Normalize
- Apply Histogram Equalization

Figure (A)

CDF(v) is the number of pixels whose value is less or equal than v. After computing CDF, you must make sure to normalize your CDF so that the range is 0 to 65535.

- The function for computing **CDF** is as below: **a** is denoted as the histogram.

```
numpy.cumsum(a, axis=None, dtype=None, out=None)
```

- The formula for **normalization** is as below:
- **cdf**$_{min}$ can obtained using cdf.min()

$$h(v) = \left(\frac{cdf(v) - cdf_{min}}{(M \times N) - 1} \times (L-1)\right)$$

cdf$_{min}$ is the minimum non-zero value of the cumulative distribution function.

M x N gives the image's number of pixels (where **M** is **width** and **N** the **height**).

L is the **number of gray levels used (65536).**

Question:

Select an appropriate method(s) and technique(s) to obtain the output image by completing the code below. Click here to download the input image.

Expected Output (Y):

Input Image (X) Output Image (Y)

Sample Code:

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

# Load the image here - 1 mark

# Compute the histogram - 2 marks

# Compute the Cumulative Distribution Function (CDF) - 1 mark

# Normalize the histogram - 5 marks
```

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

img = cv2.imread('',0)

hist1 = cv2.calcHist([img],[0], None, [256],
[0,256])

CDF = np.cumsum(hist1, axis=None,
dtype=None, out=None)

#no of gray levels
L = 65536

#total no of pixels
M,N = img.shape[:2]
total_pixels = M*N

cdf_normalized = (CDF - CDF.min()) * (L-1) /
total_pixels - 1

equalized = cv2.equalizeHist(img)
equalized_neg = cv2.bitwise_not(equalized)
_,binaryimg = cv2.threshold
(equalized_neg,100,255,cv2.THRESH_BINAR
Y)

result = np.hstack((img, binaryimg))
plt.imshow(result,cmap='gray')
plt.show()
```