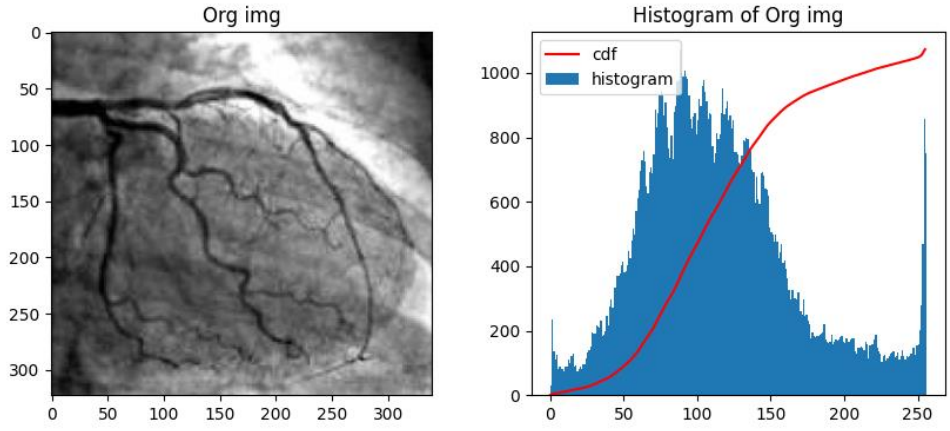
1. الف) برای حل این سوال ابتدا با استفاده از کد زیر نمودار هیستوگرام و cdf تصویر اصلی را رسم میکنیم :

```
#Read image
img = cv2.imread("drive/MyDrive/DIP_EXC2/Q1/Q1_img.jpg", 0

#Calcute hist and cdf
hist,bins = np.histogram(img.flatten(),256,[0,256])
cdf = hist.cumsum()
cdf_normalized = cdf * float(hist.max()) / cdf.max()

#Show histogram & image
f, subplt = plt.subplots(1,2, figsize=(10,4))

subplt[0].imshow(img, cmap="gray")
subplt[0].set_title("Org img")
subplt[1].plot(cdf_normalized, color = 'b')
subplt[1].hist(img.flatten(),256,[0,256], color = 'r')
subplt[1].legend(('cdf','histogram'), loc = 'upper left')
subplt[1].set_title("Histogram of Org img")
plt.show()
```

تصویر اصلی و نمودار هیستوگرام و cdf آن :



حال با استفاده از کد زیر نرمال سازی هیستوگرام را بدون استفاده از cv2.equalizeHist انجام میدهیم .

ابتدا با استفاده از دو پارامتر height و width که به اندازه ی ارتفاع و عرض تصویر هستند، هیستوگرام را به دست می آوریم:

```
for i in range(width):
    for j in range(height):
        g = img[j,i]
        a[g] = a[g]+1
```

سپس با دستور زیر نرمال سازی انجام داده و مقادیر به دست آمده را ضربدر بزرگترین مقدار روشنایی پیکسل در تصویر که در این تصویر 255 می باشد کرده و round آن را به دست می آوریم :

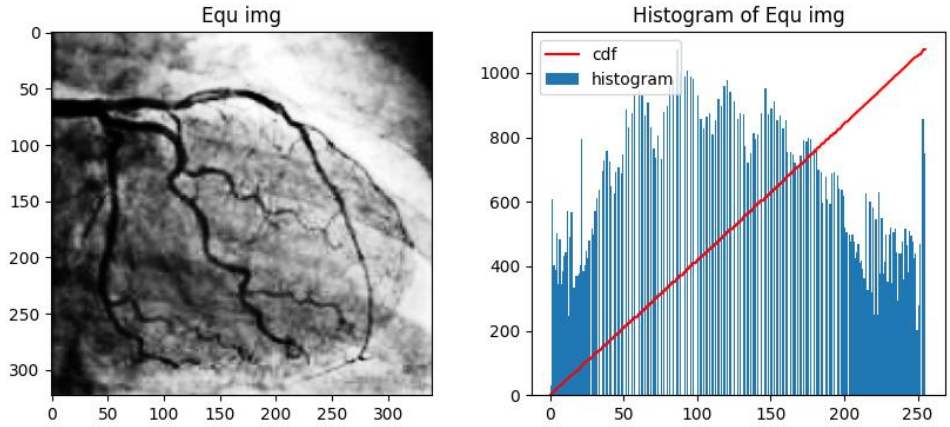
```
height*width) )/tmp = 1.0
 for i in range(256):
    for j in range(i+1):
       b[i] += a[j] * tmp;
b[i] = round(b[i] * 255);
```

کد کلی این قسمت به صورت زیر است :

```
#define a & b
a = np.zeros((256,),dtype=np.float16)
b = np.zeros((256,),dtype=np.float16)

height,width=img.shape

#calcute histogram
for i in range(width):
    for j in range(height):
        g = img[j,i]
        a[g] = a[g]+1


#histogram equalization
tmp = 1.0/(height*width)
b = np.zeros((256,),dtype=np.float16)

for i in range(256):
    for j in range(i+1):
        b[i] += a[j] * tmp;
    b[i] = round(b[i] * 255);

#b now contains the equalized histogram
b=b.astype(np.uint8)
```

```
#Re-map values from equalized histogram into the image
for i in range(width):
    for j in range(height):
        g = img[j,i]
        img[j,i]= b[g]

equ_img = img

#Calcute hist and cdf
hist,bins = np.histogram(equ_img.flatten(),256,[0,256])
cdf = hist.cumsum()
cdf_normalized = cdf * float(hist.max()) / cdf.max()

#Show histogram & image
f, subplt = plt.subplots(1,2, figsize=(10,4))

subplt[0].imshow(equ_img, cmap="gray")
subplt[0].set_title("Equ img")
subplt[1].plot(cdf_normalized, color = 'b')
subplt[1].hist(equ_img.flatten(),256,[0,256], color = 'r')
subplt[1].legend(('cdf','histogram'), loc = 'upper left')
subplt[1].set_title("Histogram of Equ img")
plt.show()
```

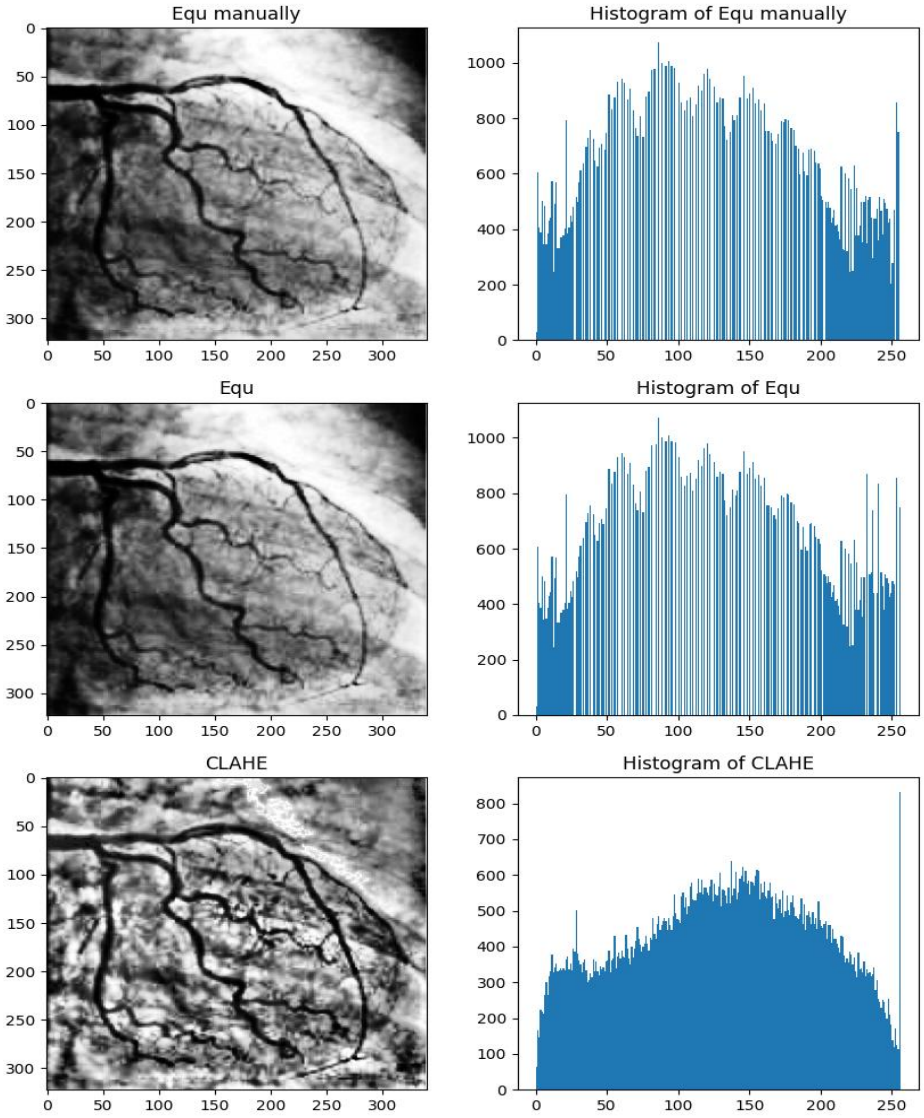
تصویر ینکواخت شده با استفاده از الگوریتم بالا (دستی) و نمودار هیستوگرام و cdf آن :



ب) با استفاده از دو دستور cv2.equalizeHist و cv2.createCLAHE هیستوگرام این تصویر را به دست آورده و هر سه تصاویر نتیجه شده و هیستوگرام آن ها را با هم مقایسه میکنیم :

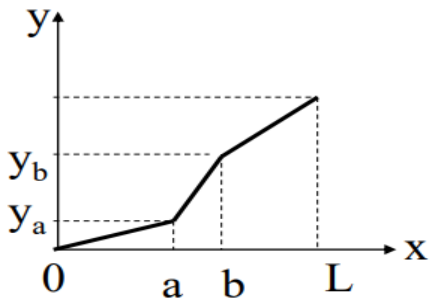
```
#Histogram Equalization
equ = cv2.equalizeHist(img)

#Create a CLAHE
clahe = cv2.createCLAHE(clipLimit=30, tileGridSize=(8,8))
cl = clahe.apply(img)

#Show histogram & image
f, subplt = plt.subplots(3,2, figsize=(10,14))

subplt[0,0].imshow(equ_img, cmap="gray")
subplt[0,0].set_title("Equ manually")
subplt[0,1].hist(equ_img.flatten(),256,[0,256])
subplt[0,1].set_title("Histogram of Equ manually")

subplt[1,0].imshow(equ, cmap="gray")
subplt[1,0].set_title("Equ")
subplt[1,1].hist(equ.flatten(),256,[0,256])
subplt[1,1].set_title("Histogram of Equ")

subplt[2,0].imshow(cl, cmap="gray")
subplt[2,0].set_title("CLAHE")
subplt[2,1].hist(cl.flatten(),256,[0,256])
subplt[2,1].set_title("Histogram of CLAHE")
plt.show()
```

تصاویر و هیستوگرام های نتیجه شده :



2. برای هر قسمت contrast تصویر را با توجه به نمودار و فرمول های آن محاسبه میکنیم. برای پیکسل هایی که intensity آن ها از 0 تا a هست را به l*x نگاشت میکنیم . قسمت a تا b را به Ya + (a - m* (x نگاشت میکنیم . در غیر این صورت به n* Yb + (b - (x نگاشت میشوند . (آلفا همان l ، بتا همان m و گاما همان n می باشد.)

$$y=\begin{cases}\alpha x & 0\le x<a\\ \beta(x-a)+y_a & a\le x<b\\ \gamma(x-b)+y_b & b\le x<L\end{cases}$$



مقادیر پارامتر ها به صورت زیر است :

a = 40 , Ya = 20 , b = 190 , Yb = 210 , l = 0.5, m = 2 , n = 0.8

مقادیر پارامتر ها را برداری (Vectorize)کرده و روی تصویر اعمال میکنیم :

Intensity_vec = np.vectorize(Intensity)

contrast_img = Intensity_vec(img, a, Ya, b, Yb, l, m, n)

کد کلی این سوال به صورت زیر است :

```
#Read image
img = cv2.imread("drive/MyDrive/DIP_EXC2/Q2/Q2_img.jpg")

#Map input intensity level to output intensity level
def Intensity(x, a, Ya, b, Yb, l, m, n):

    if (0 <= x and x <= a):
        return l* x
    elif (a < x and x <= b):
        return m* (x - a) + Ya
    else:
        return n* (x - b) + Yb

#Define parameters:
a = 40
Ya = 20
b = 190
Yb = 210
l = 0.5
m = 2
n = 0.8

#Vectorize the function
Intensity_vec = np.vectorize(Intensity)
#Apply contrast stretching
contrast_img = Intensity_vec(img, a, Ya, b, Yb, l, m, n)
#Show image
horizontal_concat = np.concatenate((img, contrast_img), axis=1)
cv2_imshow(horizontal_concat)
```

تصویر اصلی(سمت چپ) و تصویر نتیجه شده(سمت راست) :

3.الف) برای اینکه از دستورات مستقیم openCV استفاده نکنیم . ماسک Laplacian را می نویسیم و با دستور cv2.filter2D به تصویر اعمال میکنیم :

```
L = np.array([[0,-1,0],[-1, 4, -1],[0, -1, 0]])

laplacian_img = cv2.filter2D(img, -1, L)
```

سپس برای اینکه تصویر sharp شده به دست آوریم .تصویری که روی آن laplacian اعمال کردیم را با تصویر اصلی جمع میکنیم :

```
laplacian_img  + Sharp_img = img
```

کد این قسمت به صورت زیر است :

```
#Read image
img = cv2.imread("drive/MyDrive/DIP_EXC2/Q3/Q3_img.jpg")

#Laplacian Mask
L = np.array([[0,-1,0],[-1, 4, -1],[0, -1, 0]])
laplacian_img = cv2.filter2D(img, -1, L)

#Sharping Mask
Sharp_img = img + laplacian_img

#Show image
horizontal_concat = np.concatenate((img, laplacian_img, Sharp_img), axis=1)
cv2_imshow(horizontal_concat)

#Save image
cv2.imwrite("drive/MyDrive/DIP_EXC2/Q3/Sharp_img.jpg", Sharp_img)
cv2.imwrite("drive/MyDrive/DIP_EXC2/Q3/Q3_A_img.jpg", horizontal_concat)
```

تصاویر به ترتیب از چپ به راست : تصویر اصلی، تصویر laplacian، تصویر sharp شده .

ب) در این سوال هم برای اینکه از دستورات مستقیم openCV استفاده نکنیم . ماسک GaussianBlur را می نویسیم و با دستور cv2.filter2D به تصویر اعمال میکنیم تا تصویر blur شده را به دست آوریم :

L = 1/16* (np.array([[1, 2, 1],[2, 4, 2],[1, 2, 1]]))

blur_img= cv2.filter2D(img, -1, L)

حال با توجه به فرمول های : Unsharp filtering

Unsharp masking

- Subtract a blurred version of an image from the image itself

$$g_{mask}(x,y) = f(x,y) - \bar{f}(x,y)$$

- f(x,y) : The image, $\bar{f}$(x,y): The blurred image

$$g(x,y) = f(x,y) + k * g_{mask}(x,y) \qquad ,k = 1$$

ابتدا تصویر blur شده را از تصویر اصلی کم میکنیم :

blur_img - Gmask = img

سپس تصویر به دست آمده از مرحله ی قبل را با تصویر اصلی جمع میکنیم تا تصویر Unsharp را به دست آوریم :

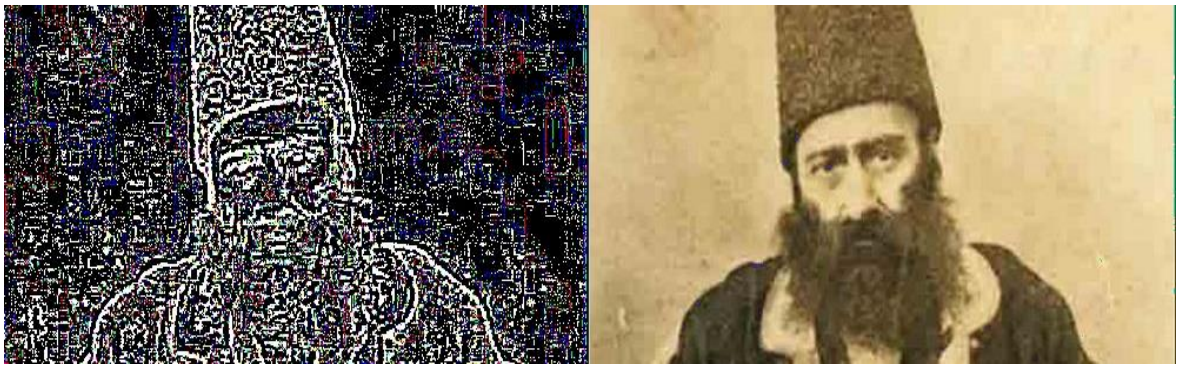
Unsharp_img = img + Gmask

کد این قسمت به صورت زیر است :

```
#GaussianBlur Mask
L = 1/16* (np.array([[1, 2, 1],[2, 4, 2],[1, 2, 1]]))
blur_img= cv2.filter2D(img, -1, L)

#GMask
Gmask = img - blur_img

#UnSharp Mask
Unsharp_img = img + Gmask

#Show image
horizontal_concat1 = np.concatenate((img, blur_img), axis=1)
horizontal_concat2 = np.concatenate((Gmask, Unsharp_img), axis=1)
cv2_imshow(horizontal_concat1)
print("Images from left to right : {original, GaussianBlur}")
cv2_imshow(horizontal_concat2)
print("Images from left to right : {Gmask, UnSharp}")
```

تصویر اصلی(سمت چپ) و تصویر blur شده (سمت راست):

تصویر Gmask(سمت چپ) و تصویر Unsharp(سمت راست):



در آخر با استفاده از کد زیر نتیجه قسمت الف و ب را باهم مقایسه میکنیم :

```
horizontal_concat = np.concatenate((Sharp_img, Unsharp_img), axis=1)

#Convert to GrayScale
horizontal_concat_Gray = cv2.cvtColor(horizontal_concat,cv2.COLOR_BGR2GRAY)

#Show image
cv2_imshow(horizontal_concat)
print("Images from left to right : {Sharp, Unsharp}\n")
print("GrayScale:\n")
cv2_imshow(horizontal_concat_Gray)
print("Images from left to right : {Sharp, Unsharp}\n")
```

تصویر Sharp شده (سمت چپ) و تصویر Unsharp فیلترینگ(سمت راست) :

در سطوح خاکستری :