INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SÃO PAULO Campus Birigui	**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA** **Campus Birigui** **Bacharelado em Engenharia de Computação**
Disciplina: Processamento Digital de Imagens	Transformações de Intensidade
Professor: Prof. Dr. Murilo Varges da Silva	**Data:** 01/09/2023
Nome do aluno: Leonardo Reneres dos Santos	**Prontuário:** BI3009131

- **Utilizar as imagens Fig 3.8 e enhance-me.gif disponíveis no Moodle**
- **Aplicar a transformação logarítmica, testar vários valores para o parâmetro c "s = c log (1 + r)"**
- **Aplicar a transformação de potência (gama), testar vários valores para o parâmetro γ e c=1 "s = crγ"**
- **Implemente a representação de cada plano de bits das imagens**
- **Implementar a equalização do histograma**
- **Elaborar relatório explicando a implementação de cada transformação e qual foi o efeito na imagem.**

Respostas:

```
from PIL import Image

import numpy as np

from numpy import asarray

import matplotlib.pyplot as plt

import cv2



def histograma(imagem):
```

```
    img = imagem.convert('L') # Converta para escala de cinza
    img_data = np.array(img)

    # Calcule o histograma da imagem
    hist, _ = np.histogram(img_data.flatten(), 256, [0, 256])

    # Calcule a função de distribuição cumulativa (CDF)
    cdf = hist.cumsum()
    cdf_normalized = cdf * hist.max() / cdf.max()

    # Aplique a equalização de histograma
    cdf_m = np.ma.masked_equal(cdf, 0)
    cdf_m = (cdf_m - cdf_m.min()) * 255 / (cdf_m.max() - cdf_m.min())
    cdf = np.ma.filled(cdf_m, 0).astype('uint8')
    equalized_img_data = cdf[img_data]

    # Exiba a imagem equalizada
    equalized_img = Image.fromarray(equalized_img_data)

    return equalized_img


def main():

   # im = np.array(Image.open('fig.tif'))
```

```
#im2 = np.array(Image.open('enhance-me.gif'))

im= Image.open('enhance-me.gif')

imt = im.copy()

img = im.copy()

ime = histograma(im)




im_data = np.array(imt)

#im2_data = im2.copy()


im_gama = np.array(img)


# Defina os valores de c para testar

t = 30

g = 10

c=1

t_gama = c*im_gama**g

t_log = t * np.log(1 + im_data)


bit_planes = []

for i in range(8):

    bit_plane = (im_data >> i) & 1

    bit_planes.append(bit_plane)


for i, bit_plane in enumerate(bit_planes):
```

```
        if bit_plane.max() == 0:
            bit_plane = np.zeros_like(bit_plane)
        else:
            bit_planes[i] = (bit_plane/bit_plane.max()) * 255


    fig = plt.figure()
    plt1 = fig.add_subplot(431)
    plt2 = fig.add_subplot(432)
    plt3 = fig.add_subplot(433)
    plt4 = fig.add_subplot(434)
    plt5 = fig.add_subplot(435)
    plt6 = fig.add_subplot(436)
    plt7 = fig.add_subplot(437)
    plt8 = fig.add_subplot(438)
    plt9 = fig.add_subplot(439)
    plt10 = fig.add_subplot(4,3,10)
    plt11 = fig.add_subplot(4,3,11)
    plt12 = fig.add_subplot(4,3,12)



    plt1.set_title('Original')
    plt2.set_title('Transformada Logaritmica')
    plt3.set_title('Gama')
    plt4.set_title('Bit Plane 1')
    plt5.set_title('Bit Plane 2')
```

```
plt6.set_title('Bit Plane 3')

plt7.set_title('Bit Plane 4')

plt8.set_title('Bit Plane 5')

plt9.set_title('Bit Plane 6')

plt10.set_title('Bit Plane 7')

plt11.set_title('Bit Plane 8')

plt12.set_title('Equalize')




plt1.imshow(im, cmap='gray', vmin=0, vmax=255)

plt2.imshow(t_log, cmap='gray', vmin=0, vmax=255)

plt3.imshow(t_gama, cmap='gray', vmin=0, vmax=255)

plt4.imshow(bit_planes[0], cmap='gray', vmin=0, vmax=255)

plt5.imshow(bit_planes[1], cmap='gray', vmin=0, vmax=255)

plt6.imshow(bit_planes[2], cmap='gray', vmin=0, vmax=255)

plt7.imshow(bit_planes[3], cmap='gray', vmin=0, vmax=255)

plt8.imshow(bit_planes[4], cmap='gray', vmin=0, vmax=255)

plt9.imshow(bit_planes[5], cmap='gray', vmin=0, vmax=255)

plt10.imshow(bit_planes[6], cmap='gray', vmin=0, vmax=255)

plt11.imshow(bit_planes[7], cmap='gray', vmin=0, vmax=255)

plt12.imshow(ime, cmap='gray', vmin=0, vmax=255)


plt.show()
```

```
    return 0


if __name__ == '__main__':
    main()
```