National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Computing

EE 433: Digital Image Processing

Name: Amal Saqib

CMS: 282496

Class: BSCS 9C

Lab 3: Basic Image Processing

Date: 27th September 2021

Time: 2.00Pm to 5.00Pm

Instructor: Dr. Imran Malik

Lab 3

Basic Image Processing

Task #1: Image Binarization using a predefined global threshold.

Take an RGB image (preferably from the provided ones) and convert it to binarized form (in 0/1 form) by defining a single global threshold. Repeat the experiment with the three provided images and identify why a single global binarization threshold may not be applicable in a wide variety of application scenarios.
Note :_ you have to perform task 1 on following images B1,B2,B3

Refer output of B3.jpg
Not all images can be binarized using the same threshold

CODE

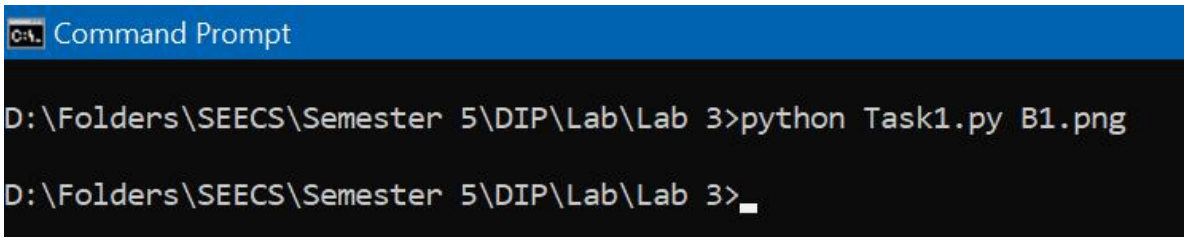
```
import os, sys
from PIL import Image

# decide the threshold
threshold = 164

# binarize image
def binarization(img):
    # load the pixels of the image
    pix = img.load()
    # get width and height of the input image
    width, height = img.size
    # iterate through all the pixels
    for x in range(width):
        for y in range(height):
            if pix[x, y] > threshold:
                pix[x, y] = 255
            else:
                pix[x, y] = 0
    return img

# gets all the files from the command line arguments
for infile in sys.argv[1:]:
    try:
        f, e = os.path.splitext(infile)
        # open image
        img = Image.open(infile).convert('L')
```

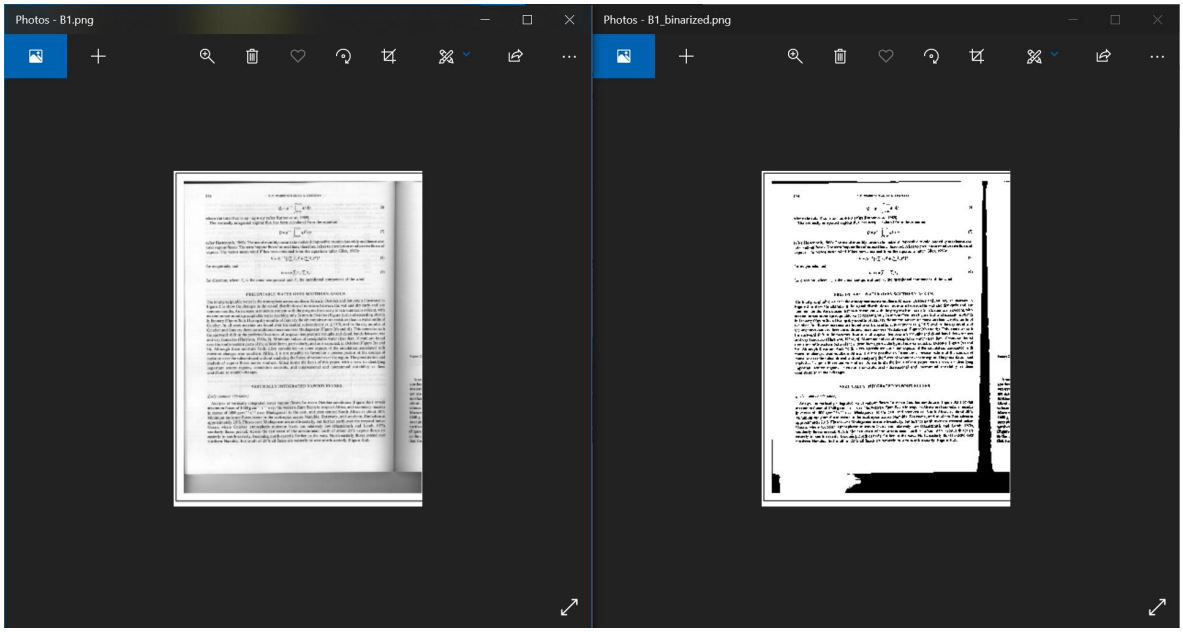
```
    binarized_img = binarization(img)

    # save binarized image
    binarized_img.save(f + '_binarized' + e)
except IOError:
    print("Binarization Error")
```

```python
import os, sys
from PIL import Image

# decide the threshold
threshold = 164

# binarize image
def binarization(img):
    # load the pixels of the image
    pix = img.load()
    # get width and height of the input image
    width, height = img.size
    # iterate through all the pixels
    for x in range(width):
        for y in range(height):
            if pix[x, y] > threshold:
                pix[x, y] = 255
            else:
                pix[x, y] = 0
    return img

# gets all the files from the command line arguments
for infile in sys.argv[1:]:
    try:
        f, e = os.path.splitext(infile)
        # open image
        img = Image.open(infile).convert('L')

        binarized_img = binarization(img)

        # save binarized image
        binarized_img.save(f + '_binarized' + e)
    except IOError:
        print("Binarization Error")
```

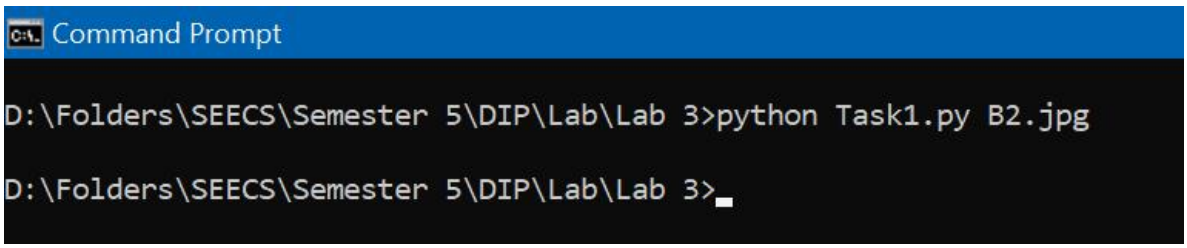
OUTPUT

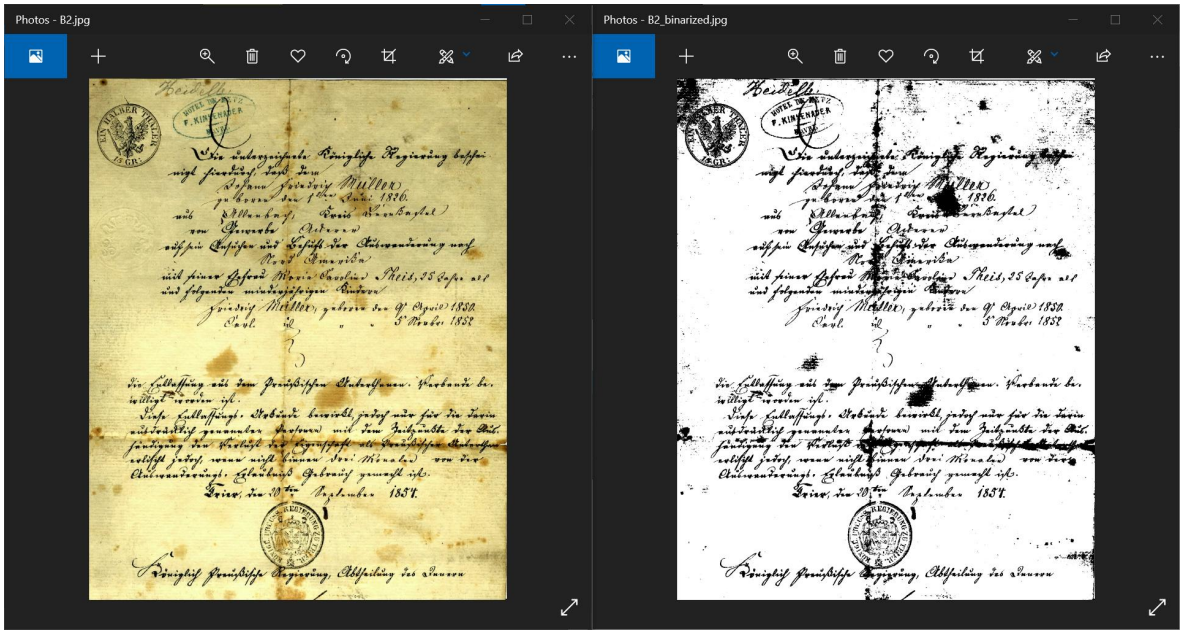
B1.png



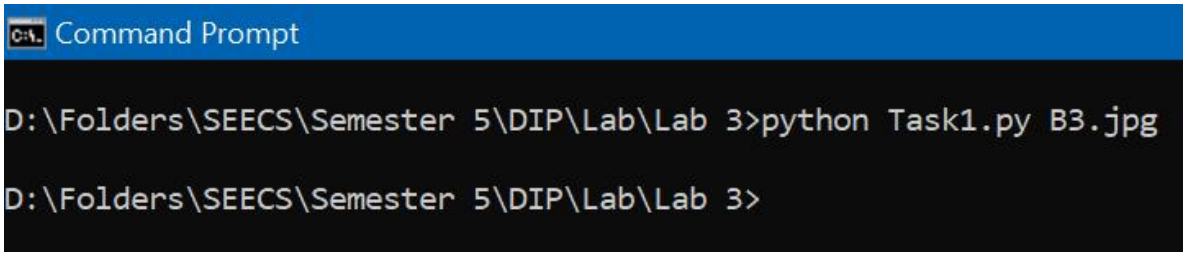


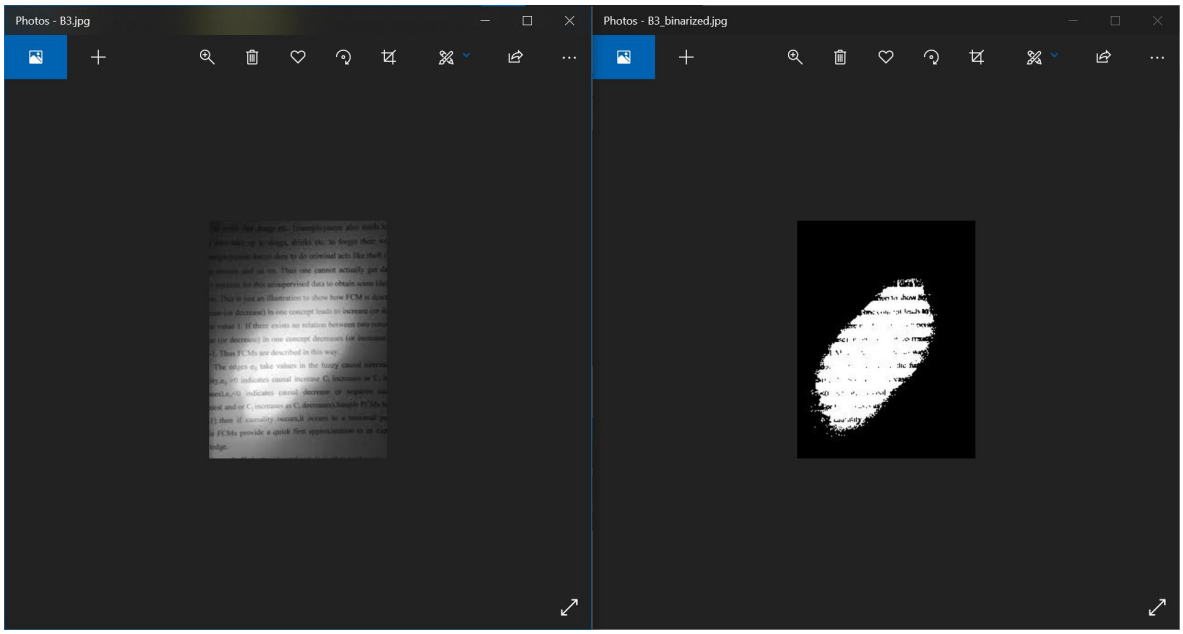
B2.jpg





B3.jpg





Task #2: Create Intensity Histogram from a Greyscale image

Hint: As discussed during the lab demo.

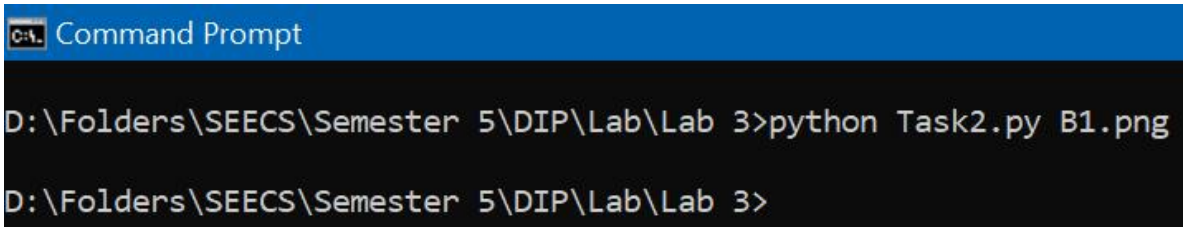
CODE

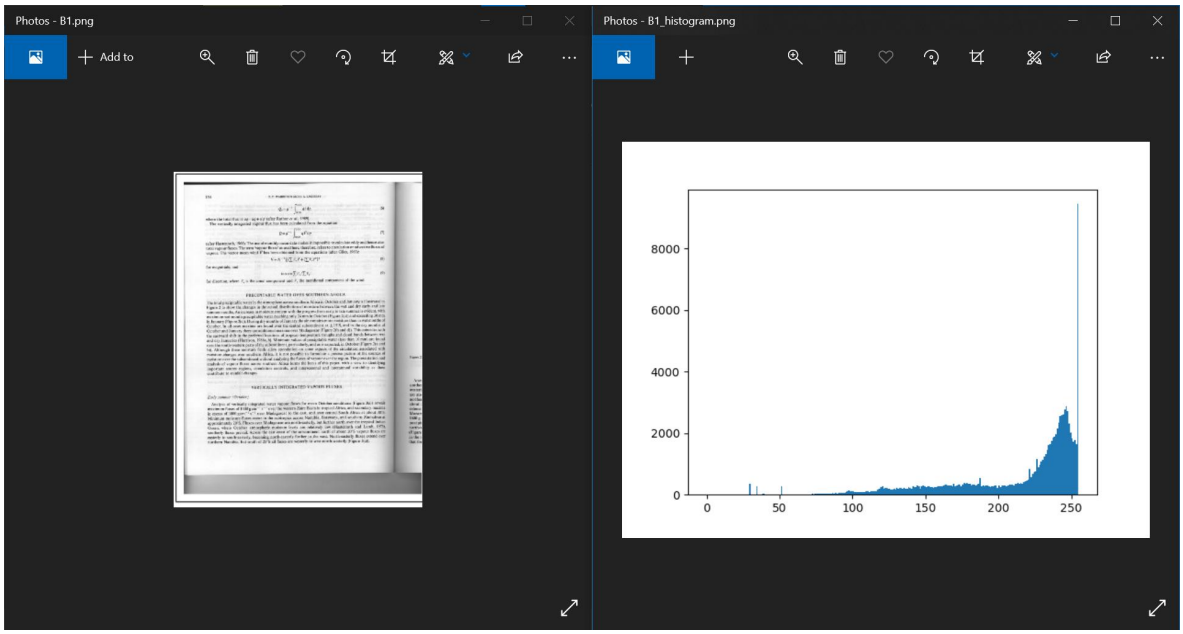
```
import os, sys
from PIL import Image
from matplotlib import pyplot as plt

# get all the pixels in an array
def histogramArray(img):
    histogram_array = []
    # load the pixels of the image
    pix = img.load()
    # get width and height of the input image
    width, height = img.size

    # iterate through all the pixels
    for x in range(width):
        for y in range(height):
            histogram_array.append(pix[x,y])
    return histogram_array

def makeHistogram(array):
    # make histogram
    plt.hist(array, bins = [i for i in range(0,256)])

# gets all the files from the command line arguments
for infile in sys.argv[1:]:
    try:
        f, e = os.path.splitext(infile)
        # open image and convert it to grayscale
        img = Image.open(infile).convert('L')

        makeHistogram(histogramArray(img))

        # save histogram
        plt.savefig(f + "_histogram" + e)

    except IOError:
        print("Error")
```

```python
import os, sys
from PIL import Image
from matplotlib import pyplot as plt

# get all the pixels in an array
def histogramArray(img):
    histogram_array = []
    # load the pixels of the image
    pix = img.load()
    # get width and height of the input image
    width, height = img.size

    # iterate through all the pixels
    for x in range(width):
        for y in range(height):
            histogram_array.append(pix[x,y])
    return histogram_array

def makeHistogram(array):
    # make histogram
    plt.hist(array, bins = [i for i in range(0,256)])

# gets all the files from the command line arguments
for infile in sys.argv[1:]:
    try:
        f, e = os.path.splitext(infile)
        # open image and convert it to grayscale
        img = Image.open(infile).convert('L')

        makeHistogram(histogramArray(img))

        # save histogram
        plt.savefig(f + "_histogram" + e)

    except IOError:
        print("Error")
```

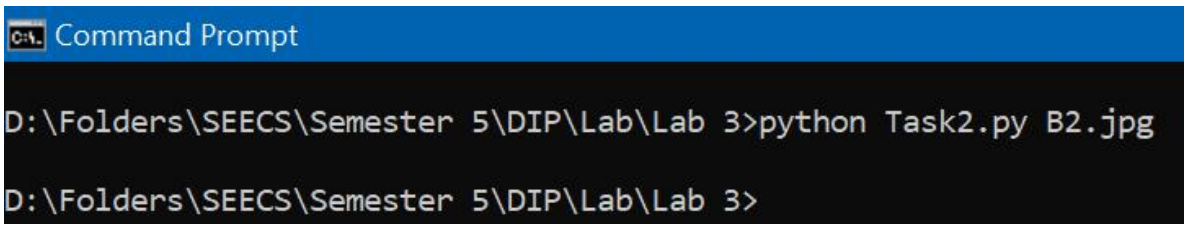
OUTPUT

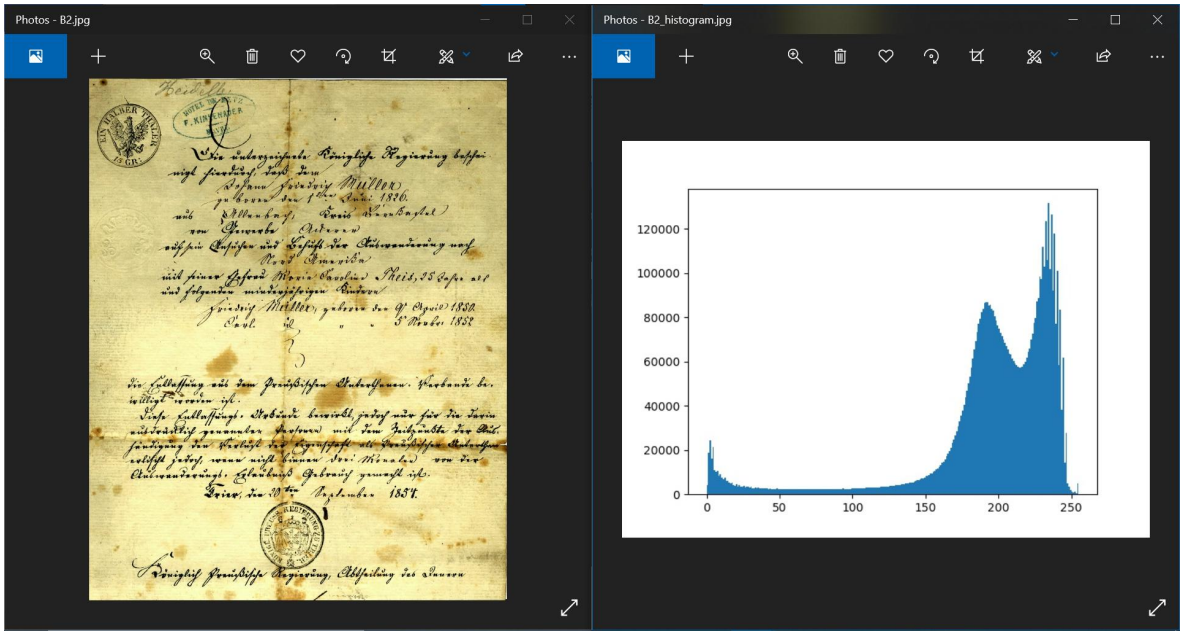
B1.png


Command Prompt
D:\Folders\SEECS\Semester 5\DIP\Lab\Lab 3>python Task2.py B1.png
D:\Folders\SEECS\Semester 5\DIP\Lab\Lab 3>

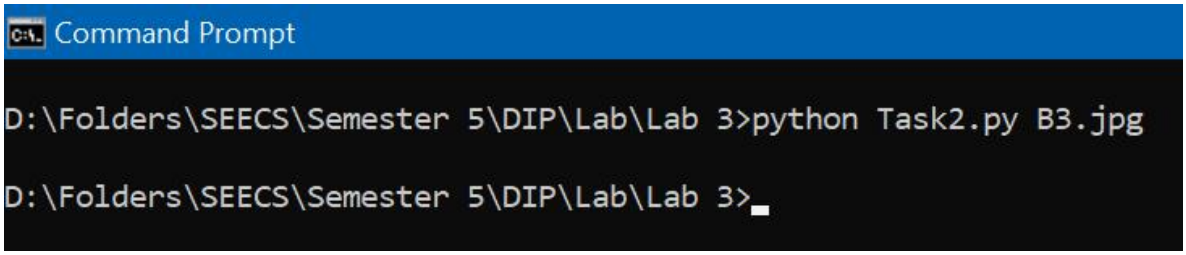

Photos - B1.png
Add to
Photos - B1_histogram.png
8000
6000
4000
2000
0
0
50
100
150
200
250

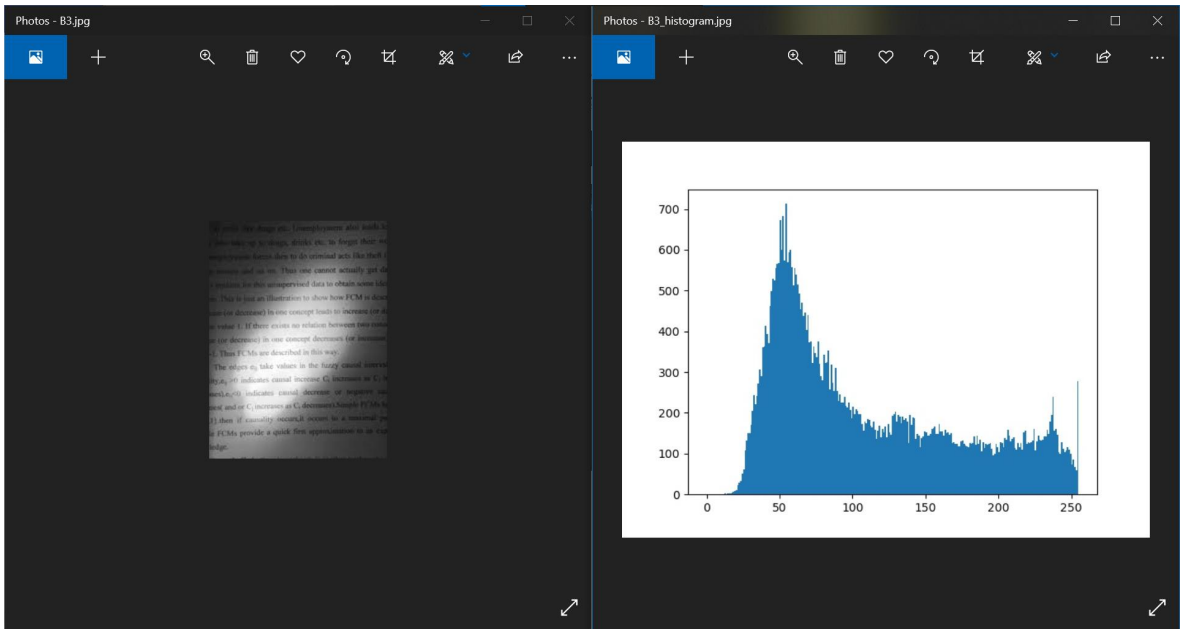
B2.jpg


Command Prompt
D:\Folders\SEECS\Semester 5\DIP\Lab\Lab 3>python Task2.py B2.jpg
D:\Folders\SEECS\Semester 5\DIP\Lab\Lab 3>



B3.jpg





Task #3: Recursive XY-cut algorithm

Hint: As discussed in the demo.
Apply it on the following image :- xycuts

CODE

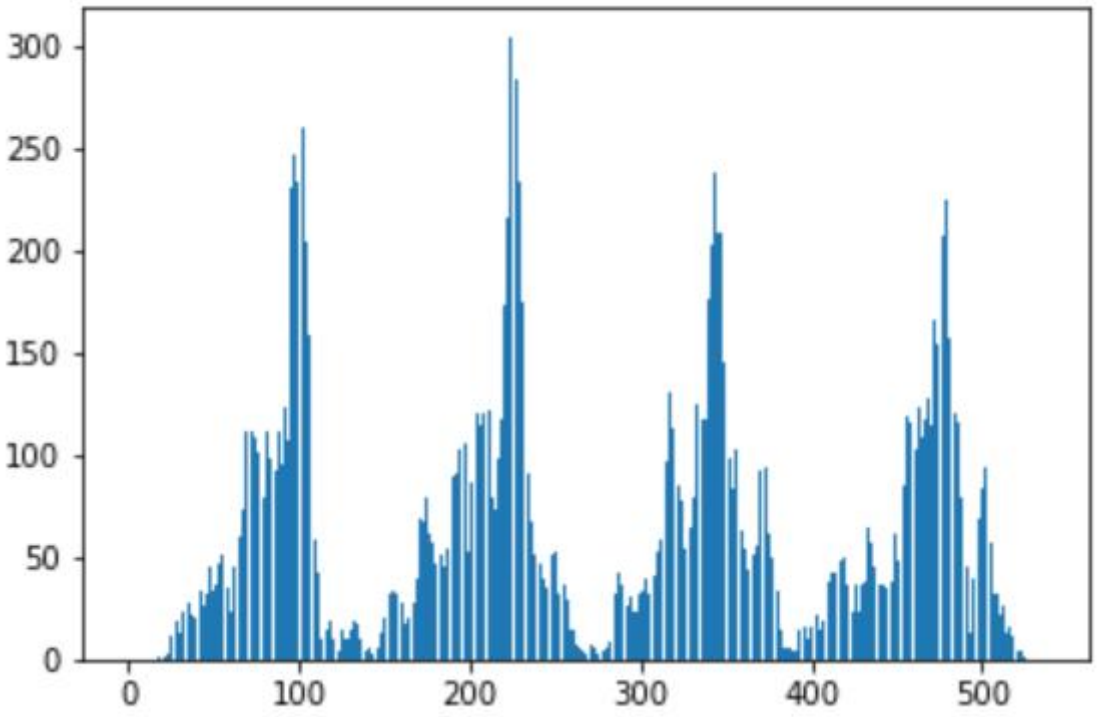
```
import os
from PIL import Image
from matplotlib import pyplot as plt

# returns the array that is used to make the histogram
def histogramArray(img):
    histogram_array = []
    # load the pixels of the image
    pix = img.load()
    # get width and height of the input image
    width, height = img.size

    # iterate through all the pixels
    for x in range(height):
        # keeps count of the number of pixels in one row
        black_pixels = 0
        for y in range(width):
            # if pixel is black, increment black_pixels
            if pix[y, x] == 0:
                black_pixels += 1
        # append the number of black_pixels in a row
        histogram_array.append(black_pixels)

    return histogram_array

# make histogram
def makeHistogram(array, img):
    plt.bar([i for i in range(0, img.height)], array)

# draw a line in rows with no or less black pixels
def makeLines(img):
    pix = img.load()
    for x in range(img.height):
        if x == 135 or x == 260 or x == 390:
            for i in range(img.width):
                pix[i, x] = 0
    return img

infile = "XY-cuts.png"
try:
    f, e = os.path.splitext(infile)
    # open image and convert it to grayscale
    img = Image.open(infile).convert('L')
```

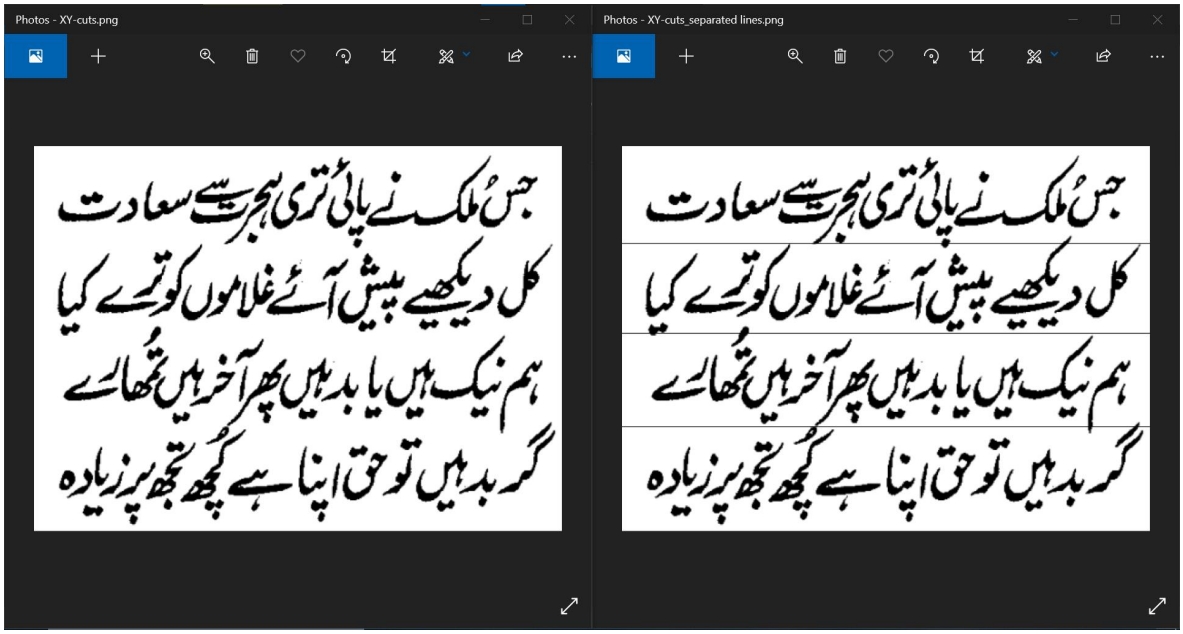
```
    makeHistogram(histogramArray(img), img)
    img = makeLines(img)

    # save histogram
    plt.savefig(f + "_histogram" + e)

    # save the new image
    img.save(f + "_separated lines" + e)

except IOError:
    print("Error")
```

```python
import os
from PIL import Image
from matplotlib import pyplot as plt

# returns the array that is used to make the histogram
def histogramArray(img):
    histogram_array = []
    # load the pixels of the image
    pix = img.load()
    # get width and height of the input image
    width, height = img.size

    # iterate through all the pixels
    for x in range(height):
        # keeps count of the number of pixels in one row
        black_pixels = 0
        for y in range(width):
            # if pixel is black, increment black_pixels
            if pix[y, x] == 0:
                black_pixels += 1
        # append the number of black_pixels in a row
        histogram_array.append(black_pixels)

    return histogram_array

```

```
26  # make histogram
27  def makeHistogram(array, img):
28      plt.bar([i for i in range(0, img.height)], array)
29
30  # draw a line in rows with no or less black pixels
31  def makeLines(img):
32      pix = img.load()
33      for x in range(img.height):
34          if x == 135 or x == 260 or x == 390:
35              for i in range(img.width):
36                  pix[i, x] = 0
37      return img
38
39  infile = "XY-cuts.png"
40  try:
41      f, e = os.path.splitext(infile)
42      # open image and convert it to grayscale
43      img = Image.open(infile).convert('L')
44
45      makeHistogram(histogramArray(img), img)
46      img = makeLines(img)
47
48      # save histogram
49      plt.savefig(f + "_histogram" + e)
50
51      # save the new image
52      img.save(f + "_separated lines" + e)
53
54  except IOError:
55      print("Error")
```

OUTPUT





Hand in

Submit a lab report containing both the code and screenshots of output.

To Receive Credit

1. By showing up on time for lab, working on the lab solution, and staying to the end of the class period, only then you can receive full credit for the lab

assignment.

2. Comment your program heavily. Intelligent comments and a clean, readable formatting of your code account for 20% of your grade.

Due date:
Check on lms