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 4: Primitive Transformations

Date: 4th October 2021

Time: 2.00Pm to 5.00Pm

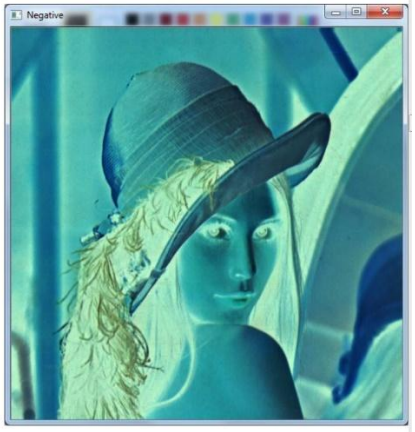
Instructor: Dr. Imran Malik

Lab 4

Primitive Transformations

Task #1: Image Negatives

Implement a function for displaying negative of an input image. Note that the function must handle binary, grayscale, and RGB images. Example of RGB negative:

From this:  To this: 

CODE

```
import os, sys
from PIL import Image

# returns the array that is used to make the histogram
def negatives(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 pixel is greyscale
            if type(pix[x,y]) == int:
                # apply the formula
                S = 255 - pix[x, y]
                # replace pixel
                pix[x, y] = S

            # if pixel is RGB
            else:
                # extract RGB from the pixel
                r, g, b = pix[x, y]
```

```
            # apply the formula
            S1 = 256 - 1 - r
            S2 = 256 - 1 - g
            S3 = 256 - 1 - b
            # replace the pixel with its negative
            pix[x, y] = (S1, S2, S3)
    return img

for infile in sys.argv[1:]:
    try:
        # split the name of the file
        f, e = os.path.splitext(infile)
        # open image
        img = Image.open(infile)

        img = negatives(img)

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

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

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

# returns the array that is used to make the histogram
def negatives(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 pixel is greyscale
            if type(pix[x,y]) == int:
                # apply the formula
                S = 255 - pix[x, y]
                # replace pixel
                pix[x, y] = S

```

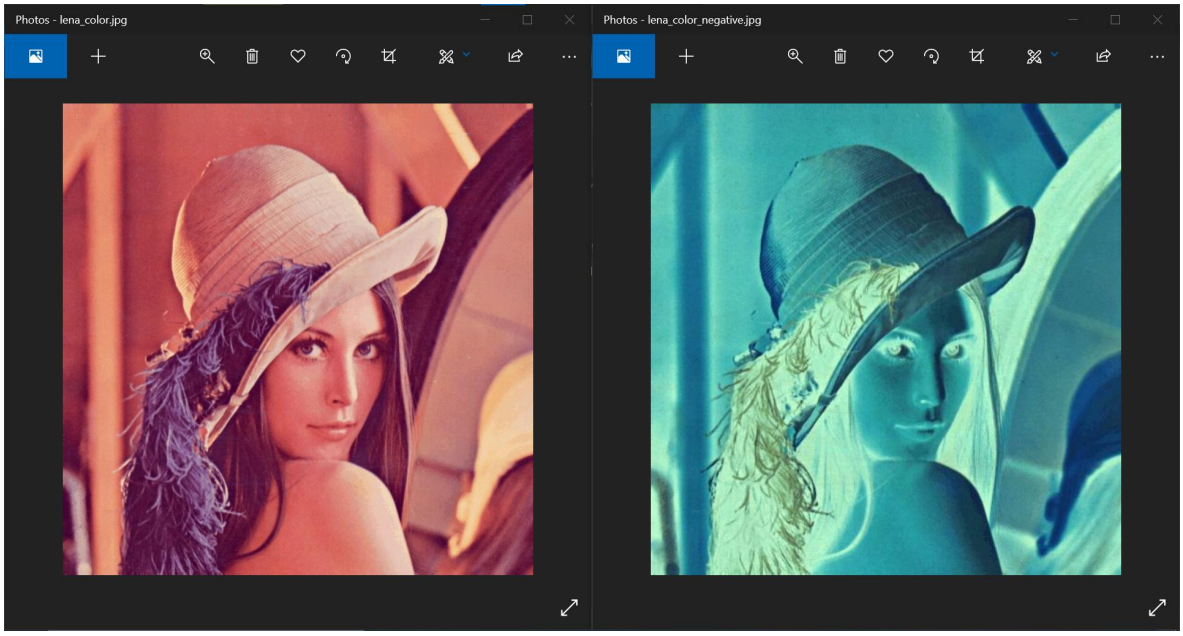
```
            # if pixel is RGB
            else:
                # extract RGB from the pixel
                r, g, b = pix[x, y]
                # apply the formula
                S1 = 256 - 1 - r
                S2 = 256 - 1 - g
                S3 = 256 - 1 - b
                # replace the pixel with its negative
                pix[x, y] = (S1, S2, S3)
    return img

for infile in sys.argv[1:]:
    try:
        # split the name of the file
        f, e = os.path.splitext(infile)
        # open image
        img = Image.open(infile)

        img = negatives(img)

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

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

RGB -

Running in cmd

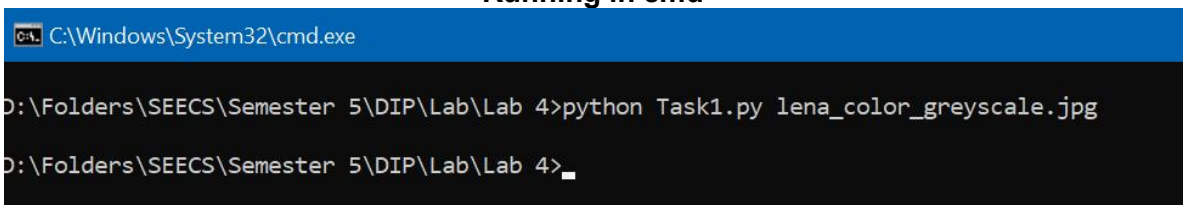
```
C:\Windows\System32\cmd.exe

D:\Folders\SEECS\Semester 5\DIP\Lab\Lab 4>python Task1.py lena_color.jpg

D:\Folders\SEECS\Semester 5\DIP\Lab\Lab 4>
```

OUTPUT

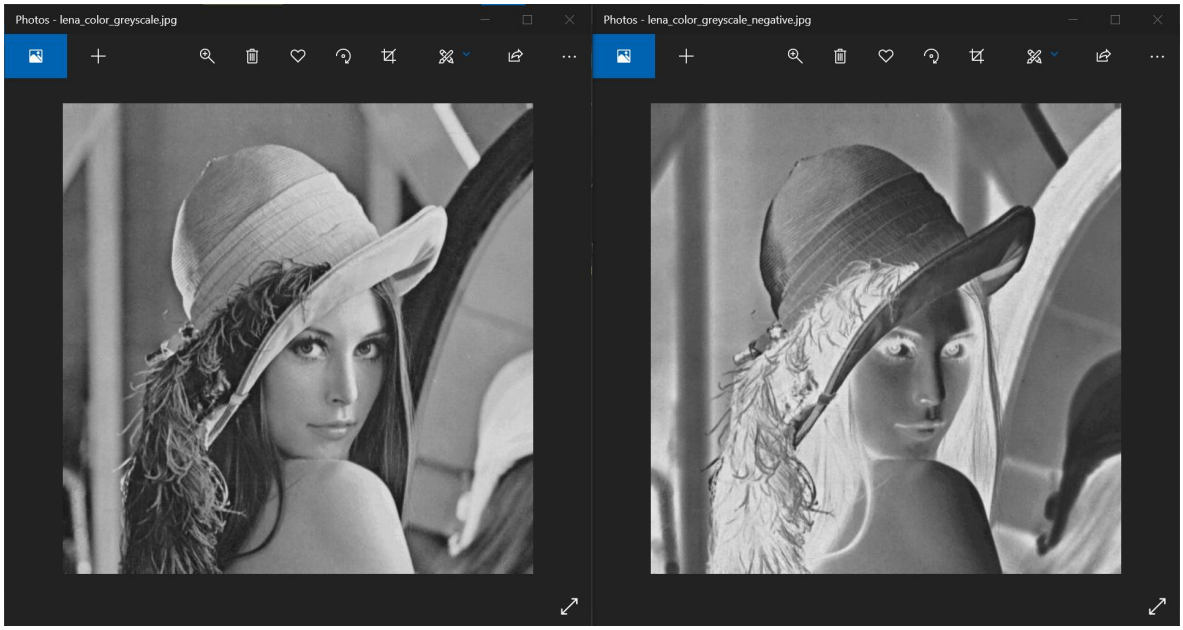


Greyscale - Using code from previous labs, I saved a greyscale copy of lena-color.jpg.

Running in cmd

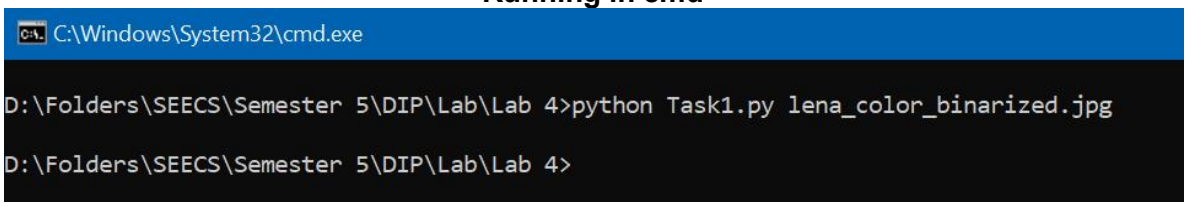


OUTPUT

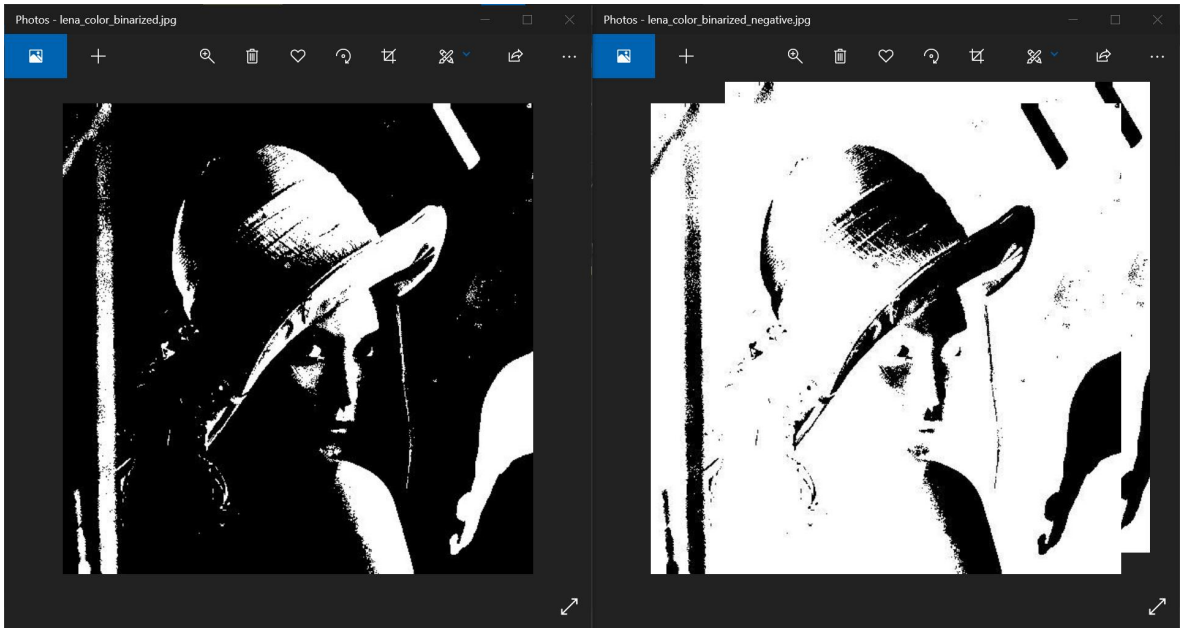


Binarized - Using code from previous labs, I saved a binarized copy of lena-color.jpg.

Running in cmd



OUTPUT



Task #2: Image Gradients

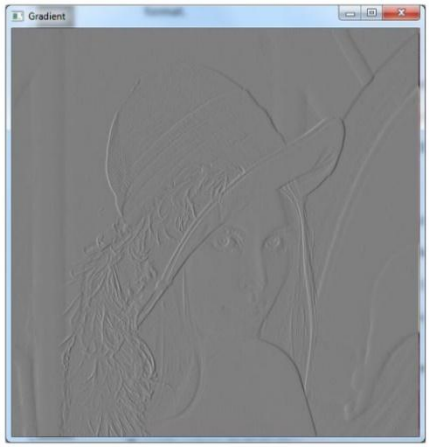
The horizontal gradient image can be used to detect vertical edges in an image. Implement a function for displaying the horizontal gradient of a grayscale image. The gradient can be approximated by forward differences:

$$I_{gradient}(x, y) = I(x + 1, y) - I(x, y)$$

Note that the gradient values can be both positive and negative! So you need to find a way to display the gradient values in the range: 0, 1, 2, …, 255. The following link can be helpful here: https://www.cis.rit.edu/people/faculty/rhody/EdgeDetection.htm

The resulting image should look something like this:

From this: 

To this: 

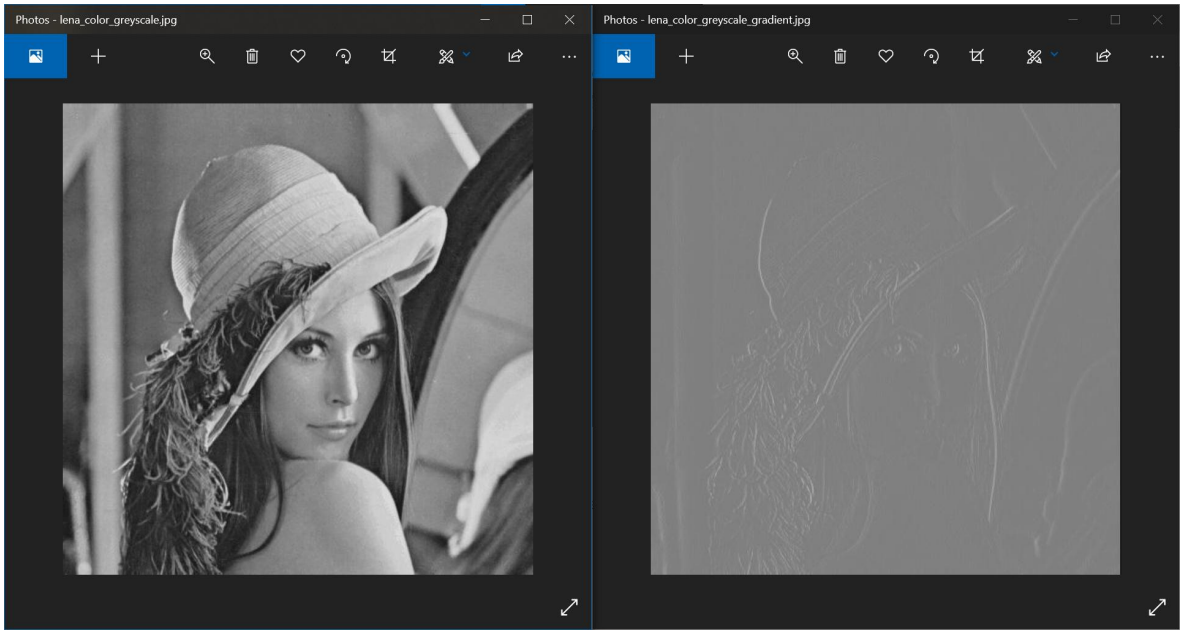
CODE

```
import os
from PIL import Image

# returns the array that is used to make the histogram
def gradient(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 - 1):
        for y in range(height):
            # apply the formula
            pix[x, y] = pix[x + 1, y] - pix[x, y]
            # scale it
            pix[x, y] = int((pix[x, y] + 255) / 510 * 255)

    # change last row of pixels to black
    for y in range(height):
        pix[width - 1, y] = 0

    return img

infile = "lena_color_greyscale.jpg"
try:
    # split the name of the file
    f, e = os.path.splitext(infile)
    # open image
    img = Image.open(infile)

    img = gradient(img)

    # save the new image
    img.save(f + "_gradient" + e)
```

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

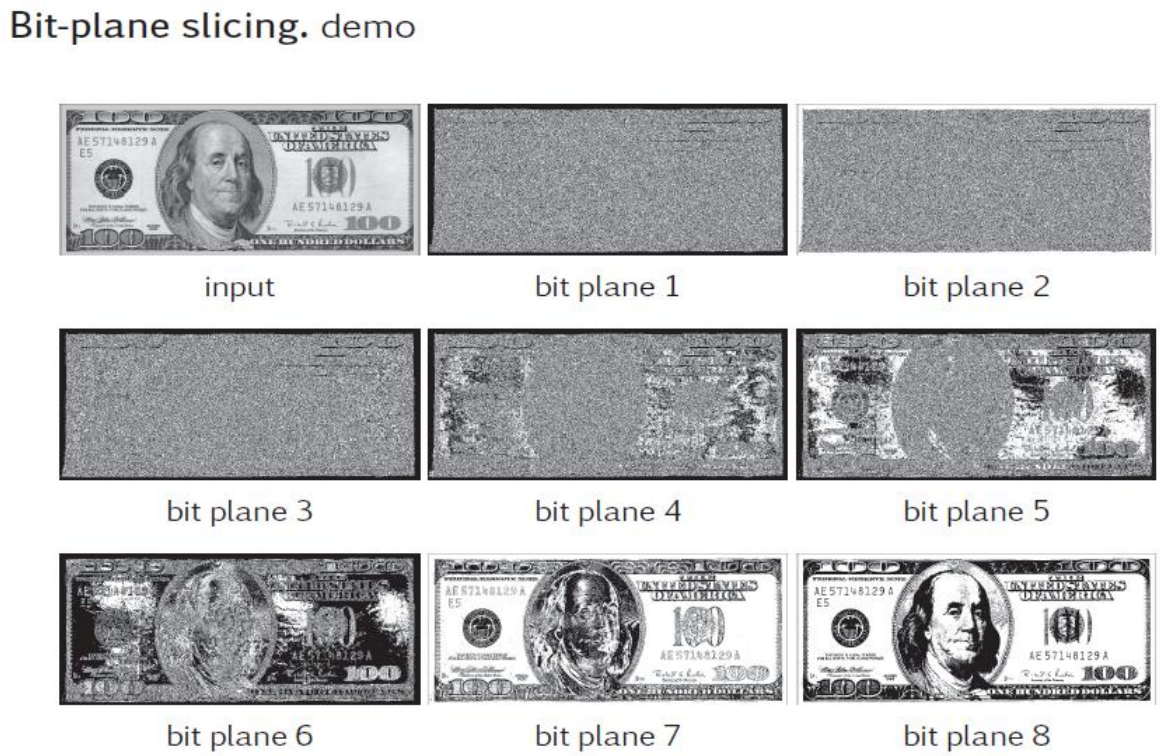
```python
import os
from PIL import Image

# returns the array that is used to make the histogram
def gradient(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 - 1):
        for y in range(height):
            # apply the formula
            pix[x, y] = pix[x + 1, y] - pix[x, y]
            # scale it
            pix[x, y] = int((pix[x, y] + 255) / 510 * 255)

    # change last row of pixels to black
    for y in range(height):
        pix[width - 1, y] = 0

    return img

infile = "lena_color_greyscale.jpg"
try:
    # split the name of the file
    f, e = os.path.splitext(infile)
    # open image
    img = Image.open(infile)

    img = gradient(img)

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

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

OUTPUT



Task #3: Bit Plane Slicing

Perform bit slicing of an 8 bit greyscale image as discussed in the lecture. Start from the least significant bit and move towards the most significant bit. You will get eight binary images of the input image as demonstrated below.



CODE

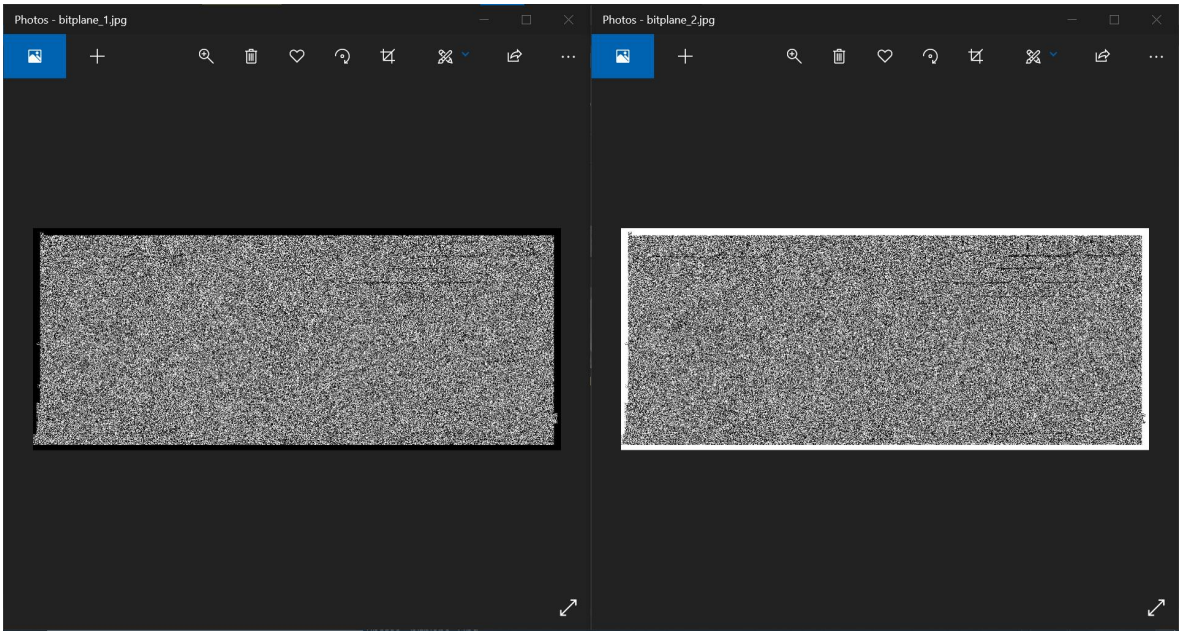
```
import os
from PIL import Image
```

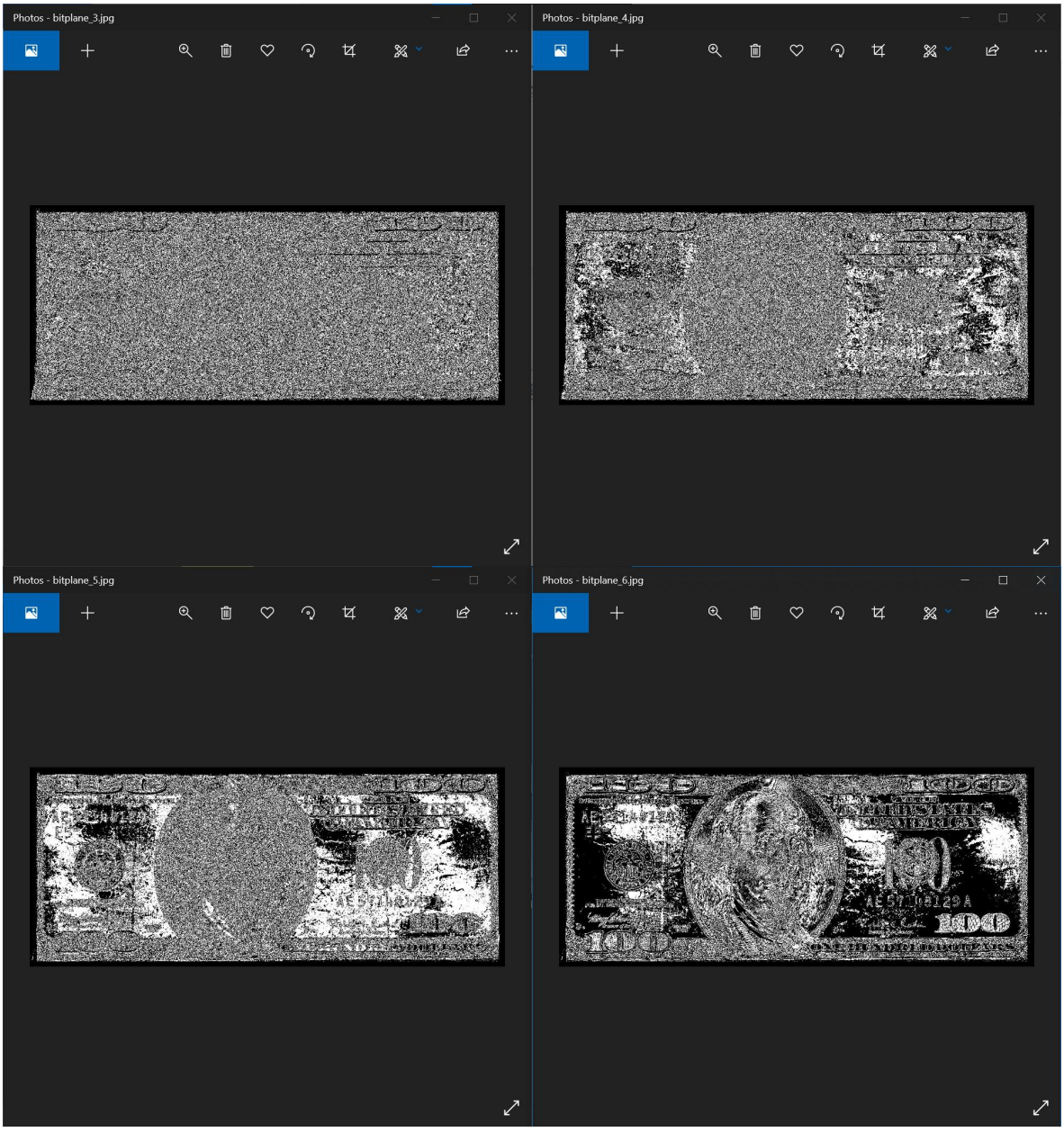
```
# convert decimal value to 8-bit binary
def toBinary(pix):
    pix = bin(pix)
    pix = pix[2:]
    while len(pix) < 8:
        pix = '0' + pix

    return pix

# returns the bit plane slice
def slicing(img, i):
    # 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):
            # find the binary equivalent of the pixel's intensity
            binary = toBinary(pix[x, y])

            # if bit is 1, make the pixel white (intensity 255)
            if int(binary[7 - i]) == 1:
                pix[x, y] = 255
            # else make the pixel black
            else:
                pix[x, y] = 0

    # return the image for that bit plane
    return img

infile = "bitplane.jpg"
try:
    # split the name of the file
    f, e = os.path.splitext(infile)

    # 8 bit slices
    for i in range(8):
        # open image
        img = Image.open(infile).convert('L')

        # call bit plane slicing for that bit
        img = slicing(img, i)

        # save the images
        img.save(f + "_" + str(i + 1) + ".jpg")

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

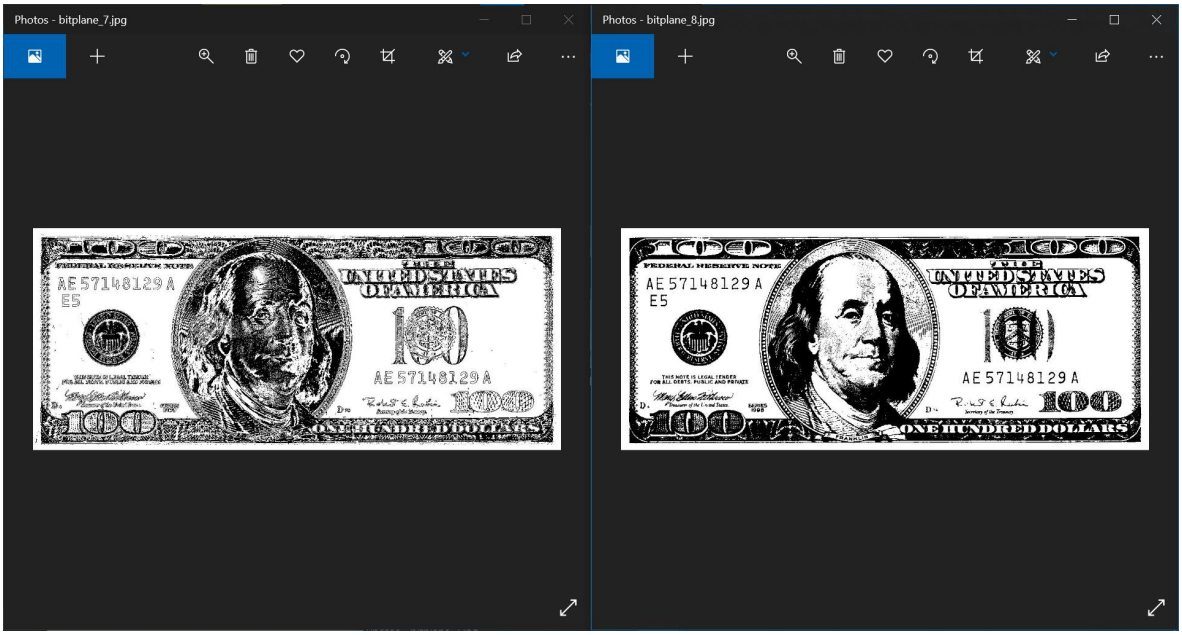
```python
import os
from PIL import Image

# convert decimal value to 8-bit binary
def toBinary(pix):
    pix = bin(pix)
    pix = pix[2:]
    while len(pix) < 8:
        pix = '0' + pix

    return pix

# returns the bit plane slice
def slicing(img, i):
    # 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):
            # find the binary equivalent of the pixel's intensity
            binary = toBinary(pix[x, y])

```

```
            # if bit is 1, make the pixel white (intensity 255)
            if int(binary[7 - i]) == 1:
                pix[x, y] = 255
            # else make the pixel black
            else:
                pix[x, y] = 0

    # return the image for that bit plane
    return img

infile = "bitplane.jpg"
try:
    # split the name of the file
    f, e = os.path.splitext(infile)

    # 8 bit slices
    for i in range(8):
        # open image
        img = Image.open(infile).convert('L')

        # call bit plane slicing for that bit
        img = slicing(img, i)

        # save the images
        img.save(f + "_" + str(i + 1) + ".jpg")

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

OUTPUT




Photos - bitplane_3.jpg
Photos - bitplane_4.jpg
Photos - bitplane_5.jpg
Photos - bitplane_6.jpg


Photos - bitplane_7.jpg
Photos - bitplane_8.jpg
AE 57148129 A
E5
THE UNITED STATES OF AMERICA
100
ONE HUNDRED DOLLARS