Image Encryption and Decryption

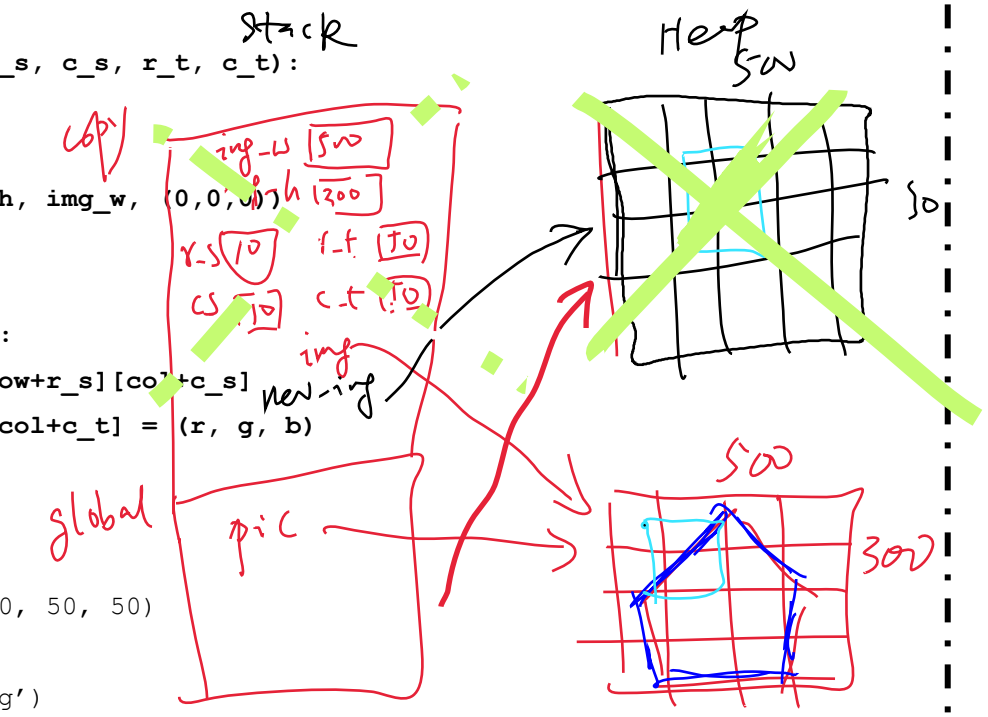
1. Image manipulation memory model

What does this code do?

```
from CSE8AImage import *
def copy_region_to_new(img, r_s, c_s, r_t, c_t):
    img_w = width(img)
    img_h = height(img)
    new_img = create_img(img_h, img_w, (0,0,0))

    for row in range(100):
        for col in range(100):
            (r, g, b) = img[row+r_s][col+c_s]
            new_img[row+r_t][col+c_t] = (r, g, b)
    return new_img

pic = load_img('arch.jpg')
copy_region_to_new(pic, 10, 30, 50, 50)
save_img(img, 'result1.jpg')
save_img(new_img, 'result2.jpg')
save_img(pic, 'result3.jpg')
```



A. This code may generate an error
B. It copies the region but there is no way to access the copied picture
C. It copies the region and result1 will show the effect
D. It copies the region and result2 will show the effect
E. It copies the region and result3 will show the effect

If we add a return new_img at the end of the copy function, what will happen?

A. This code may generate an error
B. It copies the region but there is no way to access the copied picture
C. It copies the region and result2 will show the effect
D. It copies the region and result3 will show the effect
E. None of the given answers is correct

2. Steganography

It is a way to hide information in plain sight and other than the receiver, no one can see what is hidden (unless you look really hard)

Cathy saw eight (8) airplanes penetrate skies aloft. 7 really only could keep straight.

cse 8a psa 7 rocks

Hiding information in images

The idea is to hide some information inside each pixel and use the shortcomings of human observation to make it *invisible*

Red	Green	Blue	
(39,	56,	101)	
(37,	59,	100)	

39 → 10^1, 39 → 10^0

It is better to see things and manipulate in the binary domain. MSB → 276 ← LSB $= 3 \cdot 10^2 + 9 \cdot 10^1 + 6 \cdot 10^0 = 39$

Binary representation of a number is in base 2 in contract of base 10 numbers we use every day(more in CSE 30). This is a starter for us

Convert 39 to binary.

MSB → 0b 00100111 ← LSB

2^5 2^4 2^3 2^2 2^1 2^0

$1 \cdot 2^5 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 32 + 4 + 2 + 1 = 39$

2 | 39		1
2 | 19		1
2 | 9		1
2 | 4		0
2 | 2		0
2 | 1		1
	0	

Exercise: Convert 56 and 101 to binary

use bin to verify

What is the *maximum* amount we can change a value (in decimal) by changing its two least significant digits?

A. 1 B. 2 C. 3 D. 4

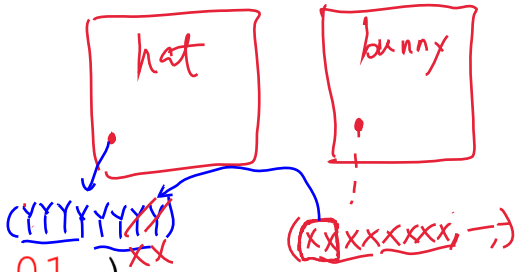
0b XXXXXXXX → 11 00

What is the *maximum* amount we can change a value (in decimal) by changing its *three* least significant digits?

A. 1 B. 3 C. 4 D. 7 E. 8

Conclusion: We can do whatever we want with the two (and probably three, maybe four?) least significant bits in each color channel without changing the visual appearance of the image. But how does this help us…?

Red	Green	Blue
(39,	56,	101)
(32,	61,	98)
(00100111,	00111000 ,	01100101)
(00100000,	**00111101 ,**	**01100010)**



How many different *colors* does8 bits per color channel allow us to represent?
A. 256 B. 256 * 3 C. 256^3 D. 8*8*8 E. None of the above

(8bits, 8bits, 8bits)

Using 2 bits per color gives us how many possible values for *each color channel*?
A. 2 B. 3 C. 4 D. 8 E. 64

(2bit, 2bit, 2bit)

If we use 2 bits per color channel, what color should (11, 00, 01) be in theory?
A. Pink(ish) B. Blue C. Green D. White E. Black

If we use 2 bits per color channel, what color should (11, 00, 01) be in Python?
A. Pink(ish) B. Blue C. Green D. White E. Black