Mirroring and Flipping

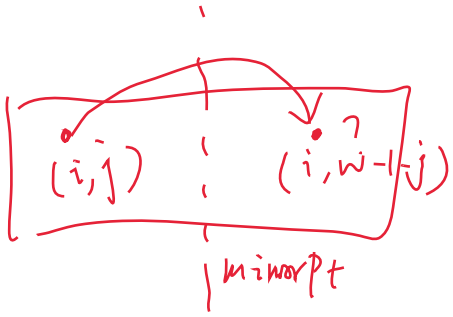
1. Mirroring

We can mirror a picture either through a horizontal axis or vertical axis. For each version, we can mirror from one side to the other (left to right, right to left, top to down, bottom to top)

right to left left to right

One of the key concept is the coordinates for pixels. Note that index goes from 0
Here is one example to mirror from left to right

```
img_w = width(img)
img_h = height(img)
mirrorPt = img_w // 2
for i in range(img_h):
  for j in range(mirrorPt):
    (rl, gl, bl) = img[i][j] #left
    img[i][img_w - j - 1] = (rl, gl, bl)#right
```



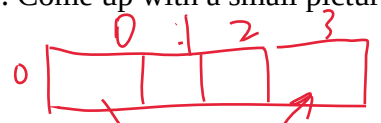
What are the coordinates we use to the left and right side for the first three iterations of the inner loop? (assume picture has a height = 50 and width = 100)

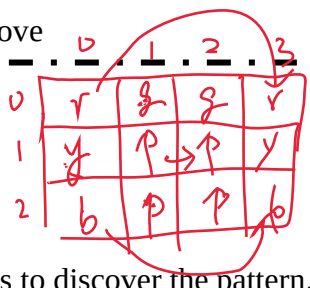
	Left Pixel	Right Pixel
A)	0, 99	99, 0
	0, 98	98, 0
	0, 97	97, 0
B)	0, 0	0, 99
	0, 1	0, 98
	0, 2	0, 97

	Left Pixel	Right Pixel
C)	0, 49	49, 0
	0, 48	48, 0
	0, 47	47, 0
D)	0, 0	49, 0
	1, 0	48, 0
	2, 0	47, 0

E. None of the above

The best way is to give yourself some example and draw a diagram
1. Come up with a small picture (pay attention to even odd width or height)



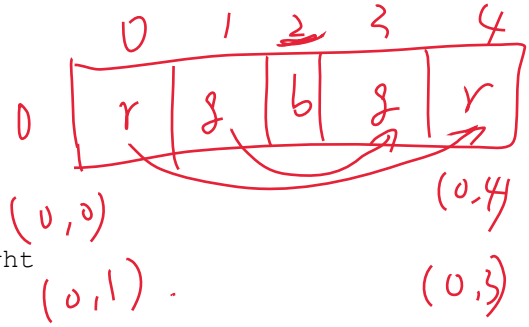


2. Think about **source** and **destination** coordinates. Write out the coordinates to discover the pattern. Usually it is about dimension//2 or dimension – row or column – 1 or something involving dimensions

What happens when this code attempts to mirror a Picture around the **vertical axis** when the Picture's width is odd (e.g. 101, or 3)?

```
img_w = width(img)
img_h = height(img)
mirrorPt = img_w // 2
for i in range(img_h):
  for j in range(mirrorPt):
    (rl, gl, bl) = img[i][j] #left
    img[i][img_w - j - 1] = (rl, gl, bl)#right
```

A. It will work fine
B. It will work fine
C. It will run, but it won't mirror correctly
D. It won't run, there will be an index out of range exception
E. It won't even run
F. None of the answers is correct



1. What are the first (x,y) coords for topP and bottomP to mirror around **horizontal** axis?

	topP	bottomP
A.	[0][0]	[0][3]
	[0][1]	[0][2]
	[1][0]	[1][3]
B.	[0][0]	[0][3]
	[1][0]	[1][3]
	[2][0]	[2][3]
C.	[0][0]	[3][0]
	[1][0]	[2][0]
	[0][1]	[3][1]
D.	[0][0]	[3][0]
	[0][1]	[3][1]
	[0][2]	[3][2]

col major

row major

E. More than one will work

2. Complete the code so the code is copied in the order specified by D

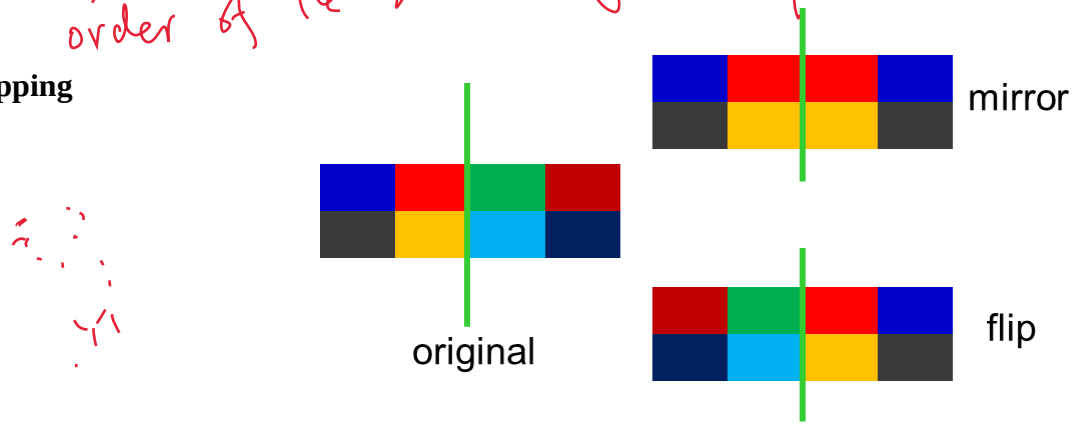
```
img_w = width(img)
img_h = height(img)
midPt = img_h//2

for i in range(_____midPt_____):
    for j in range(_____img_w_____):
        (r, g, b) = img[____i____][____j____]
        img[_img_h-i-1_][____j____] = (r, g, b)
```

col major

Row major vs Column major

order of the nested for loop

2. Flipping



Which of the following code will complete the flip

```
img_w = width(img)
img_h = height(img)
mirrorPt = img_w // 2
for i in range(mirrorPt):
  for j in range(img_w):
    (rl, gl, bl) = img[i][j]
    (rr, gr, br) = img[i][img_w - j - 1]
    #Complete this code here
```

A. `img[i][j] = (rr, gr, br)`
B. `img[i][img_w - j - 1] = (rl, gl, bl)`
C. A and then B
D. B and then A
E. More than one options are correct

What does this code do?

```
img_w = width(img)
img_h = height(img)
magic = img_w // 2
for i in range(img_h):
    countingDown = img_w - 1
    for j in range(magic):
        (r, g, b) = img[i][j]
        img[i][countingDown] = (r, g, b)
        countingDown -= 1
```

A. Copies top half into bottom half not mirrored.
B. Copies left half into right half not mirrored.
C. Mirrors around vertical axis, left into right
D. Mirrors around horizontal axis, top into bottom
E. Some other bizarre transformation

What is the size of the red box made by the following code?

```
for i in range(1, 5):
    for j in range(40, 50):
        img[i][j] = (255, 0, 0)
```

	width	height
A.	10	5
B.	9	4
C.	5	10
D.	4	9
E.	None of the above	

What are correct loops to make a black box of width w and height h?

```
def foo(img):
  img_w = width(img)
  img_h = height(img)
  magic = img_w // 2
  #Nested loop codes here
        img[row][col] = (0, 0, 0)
```

A)
```
for row in range(h+1):
    for col in range(w+1):
```

B)
```
for row in range(10, h+10):
    for col in range(20, w+20):
```

C)
```
for row in range(h):
    for col in range(w):
```

D)
```
for row in range(h, 0, -1):
    for col in range(w, 0, -1):
```

E. More than one of these