SPIS Today: Pictures and Loops, more practice!
For-loops, redux

```python
for x in range(40, 44):
    print x,
```

What does the above code print?
A. 40 41 42 43
B. 40 41 42 43 44
C. 40 44
D. Something else
Nested for-loops, redux

```python
for x in range( 40, 44 ):
    print x
    for y in range( 100, 102 ):
        print y
```

What does the above code print? (Ignore minor spacing issues.)

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>40</td>
<td>B</td>
<td>40</td>
<td>C</td>
</tr>
<tr>
<td>41</td>
<td>100</td>
<td></td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>42</td>
<td>41</td>
<td>101</td>
<td>41</td>
<td>41</td>
</tr>
<tr>
<td>43</td>
<td>101</td>
<td>41</td>
<td>41</td>
<td>41</td>
</tr>
<tr>
<td>100</td>
<td>42</td>
<td>100</td>
<td>101</td>
<td>101</td>
</tr>
<tr>
<td>101</td>
<td>102</td>
<td>101</td>
<td>42</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td>43</td>
<td>100</td>
<td>101</td>
<td>43</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100</td>
<td>43</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>43</td>
<td>101</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>101</td>
</tr>
</tbody>
</table>
The order of things...

```python
for x in range(2):
    for y in range(3):
        print x, ',', y
```

What does the above code print?

A. 0, 0  B. 0, 0  C. 0, 0  D. 0, 0
  0, 1   1, 0   0, 1   1, 0
  0, 2   2, 0   1, 0   0, 1
  1, 0   0, 1   1, 1   1, 1
  1, 1   1, 1   2, 0   0, 2
  1, 2   2, 1   2, 1   1, 2
Order matters...

```python
for x in range( getWidth( pic )):
    for y in range( getHeight( pic ) ):
        pixel = getPixel( pic, x, y )
        setColor( pixel, makeColor( 0, 0, 0 ) )
```

In what order does the above code visit the pixels?

A.  
B.  
C.  
D.  

![Diagram A](image1)

![Diagram B](image2)

![Diagram C](image3)

![Diagram D](image4)
for \( y \) in range( getHeight( \textit{pic} )):
    for \( x \) in range( getWidth( \textit{pic} )):
        \texttt{pixel} = \texttt{getPixel( \textit{pic}, x, y )}
        \texttt{setColor( \texttt{pixel}, \texttt{makeColor( 0, 0, 0 )} )}

In what order does the above code visit the pixels?
Order matters...

for y in range(getHeight(pic)):
    for x in range(getWidth(pic)):
        pixel = getPixel(pic, x, y)
        setColor(pixel, makeColor(0, 0, 0))

for x in range(getWidth(pic)):
    for y in range(getHeight(pic)):
        pixel = getPixel(pic, x, y)
        setColor(pixel, makeColor(0, 0, 0))

Do the two pieces of code above do the same thing?
A. Yes
B. No
Writing nested for loops

Fill in the code below to make the right half of the picture *pure blue*

```
for x_col in ______________________________:
    for y_row in ______________________________:
        p = getPixel( pic, x, y )
        setColor( p, __________________________ )
```
Writing nested for loops

```
for x_col in range(getWidth(pic)/2, getWidth(pic)):
    for y_row in range(getHeight(pic)):
        p = getPixel( pic, x, y )
        b = getBlue( p )
        g = getGreen( p )
        r = getRed( p )
        setColor( p, ________________________ )
```
Using if-statements

Fill in the code below to create a black border 2 pixels wide around the border of a picture. Assume the picture is at least 2 pixels wide and tall.

```python
for x_col in range(getWidth()):
    for y_row in range(getHeight()):
        if ( ________________________________):
            p = getPixel( pic, x_col, y_row )
            setColor( p, makeColor( 0, 0, 0 )
```
Copying pixels

The key to (almost) all of the image manipulation problems in lab is to copy the color value across pixels in an image. The key is figuring out which pixels to copy and where to copy them to.

Here is the generic template that you will use for almost all of these problems:

```python
for x_from in range(_______________________________):
    for y_from in range(_______________________________):
        fromPix = getPixel( pic, x_from, y_from )

        toPix = getPixel( pic, __________, __________ )
        setColor( toPix, makeColor( getRed(fromPix),
                                      getGreen(fromPix),
                                      getBlue(fromPix)) )
```
Copy right to left

Write a function that copies the right half of a picture to the left half.
Copying right to left

Write a function that copies the right half of a picture to the left half

1. Figure out the bounds of your for-loop. x_from and y_from will be your loop control variables
2. x_from and y_from are also the coordinates of the source pixel. Figure out how to represent x_to and y_to (the coordinates of the destination pixel) in terms of x_from and y_from
3. Fill in the template
Copying right to left

Write a function that copies the right half of a picture to the left half

1. Figure out the bounds of your for-loop. x_from and y_from will be your loop control variables
Copying right to left

Write a function that copies the right half of a picture to the left half

2. How does $y_{\text{to}}$ relate to $y_{\text{from}}$?
   How does $x_{\text{to}}$ relate to $x_{\text{from}}$?
for x_from in range(______________________________):
    for y_from in range(______________________________):
        fromPix = getPixel( pic, x_from, y_from )

        toPix = getPixel( pic, _________, __________ )
        setColor( toPix, makeColor( getRed(fromPix),
                                   getGreen(fromPix), getBlue(fromPix)) )