AP Computer Science A
Picture Lab 4C: Shifting Pixels

Name _____
Date _____ Period _____

Objective: To shift the pixels in a picture and produce interesting effects.

Background:
Now that we know the basics in modifying pictures, let's try some effects. Photoshop can perform a host of different effects on photos by simply modifying pixels. We can emulate those effects with some simple techniques.

One effect is to shift pixels to the left and right. This can cause interesting distortions like the before and after photos below.

We will take this one step at a time, starting with simple effects and working to more difficult ones.

Activities:

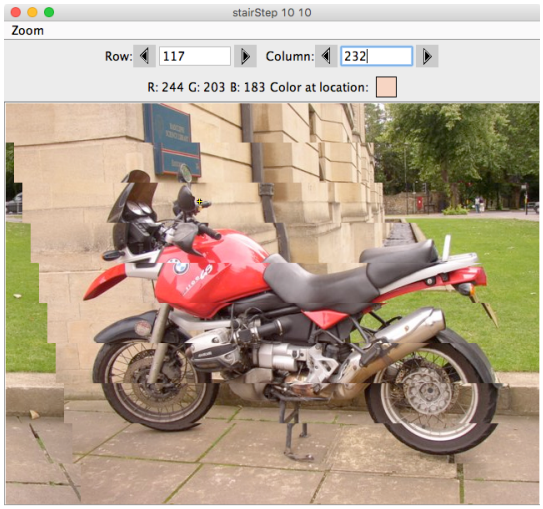
A1) **Swap Left and Right**: In this activity, we will shift pixels to the right and wrap around to the left. The left half of the picture will end up on the right side.

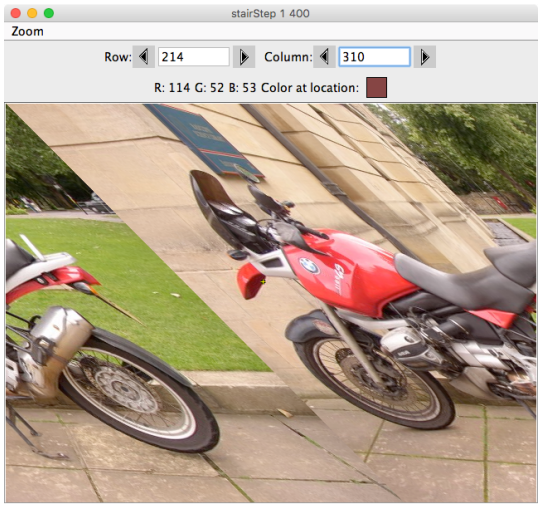
The effect is created by shifting pixels to the right and wrapping half of them around to the left. The key is to add half the width to the column, then modulus the width.

newColumn = (column + width / 2) % width

Create a new method `Picture swapLeftRight()` inside **Picture.java** to accomplish this effect.

A2) **Stair Step**: A jagged picture can be made using stair steps of shifted pixels. The left picture has 10 stair steps and each step shifts 10 pixels. The right picture has 400 stair steps and shifts each step 1 pixel. Pixels wrap around right to left.



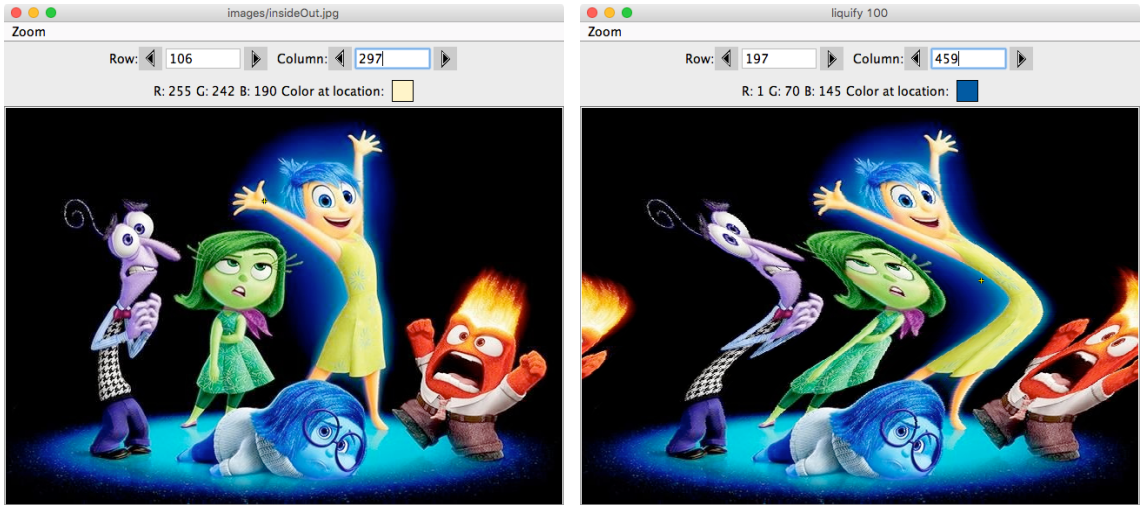


Create a new method `Picture stairStep(int shiftCount, int steps)` inside **Picture.java** to accomplish this effect. **shiftCount** is the number of pixels to shift right at each stair step. `steps` is the number of stair steps. Apply this effect to one of your own pictures.

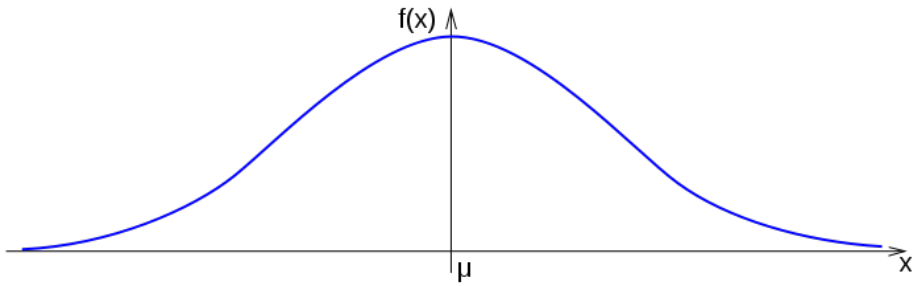
Create a **stairStep** method inside **Picture.java** using the following signature.

```
/*  <Description here>
 *  @param shiftCount  The number of pixels to shift to the right
 *  @param steps  The number of steps
 *  @return  The picture with pixels shifted in stair steps
 */
public Picture stairStep(int shiftCount, int steps)
```

A3) **Liquify**: Photoshop has a "liquify" effect which distorts a picture as you drag the mouse across it. In this activity, we will distort the horizontal center of the picture by shifting pixels horizontally.



To smooth out the distortion, a Gaussian curve is used to calculate the horizontal shift.



The formula for this curve is

$$f(x) = Ae^{-\left(\frac{(x-x_0)^2}{2d^2}\right)}$$

where A is height of the curve, x_0 is the center of the curve, and d is the standard deviation or width of the "bell". The function is applied such that x is the row, x_0 is half the height, and $f(x)$ is the pixel shift to the right (by columns). Translating this to pixel offset, in which A is **maxHeight** and d is **bellWidth**, the code is

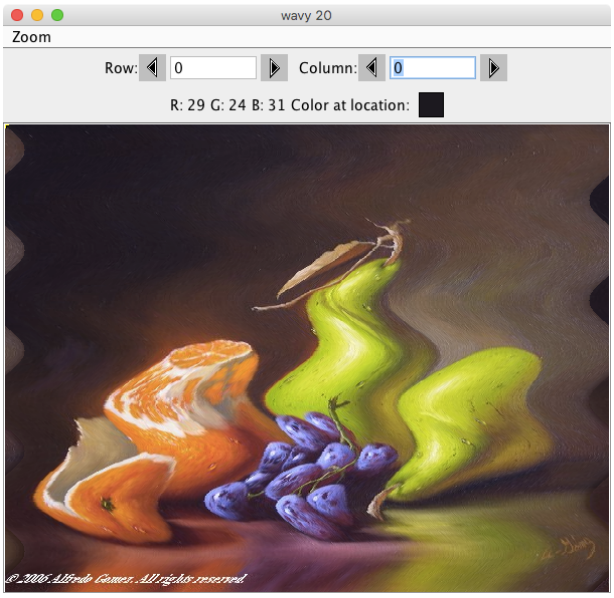
```
double exponent = Math.pow(row - height / 2.0, 2) / (2.0 * Math.pow(bellWidth, 2));

int rightShift = (int)(maxHeight * Math.exp(- exponent));
```

As pixels shift to the right, be sure to wrap the pixels around from the right to the left. Try different values for **bellWidth** and **maxHeight** to get a desirable effect. Apply this effect to one of your own pictures. Mr Greenstein suggests a **maxHeight** around 100 pixels.

Create a **liquify** method inside **Picture.java** using the following signature.

```
/*  <Description here>
 *  @param maxFactor  Max height (shift) of curve in pixels
 *  @return           Liquified picture
 */
public Picture liquify(int maxHeight)
```

A4) **Waves**: Another variation on the liquify effect is to create oscillating distortions in a picture. Instead of one Gaussian curve, the whole picture distorts left and right in a sinusoidal pattern, like the picture below on the right.



A sinusoidal function is used to calculate the horizontal shift. The formula for a sinusoid is

$$g(x) = A\sin(2\pi fx + \vartheta)$$

where A is the amplitude of the wave, f is the frequency, and ϑ is the phase shift. $g(x)$ is the pixel shift and the range is $-A$ to A.

Translate this function into code. Since the formula produces positive and negative values, be sure to handle pixels wrapping to the left and to the right. Try different values for amplitude and frequency to get a desirable effect. Apply this effect to one of your own pictures.

Create a **wavy** method inside **Picture.java** using the following signature.

```
/*  <Description here>
 *  @param amplitude  The maximum shift of pixels
 *  @return           Wavy picture
 */
public Picture wavy(int amplitude)
```