Step 1: Finding the blues

Suppose you want to know which pixels are a shade of blue, which is necessary if you want to write a program to do bluescreening. Program FindBlue contains the outline of a program to do this.

One of the methods defined in this program is called isBlue which returns true if the color passed as a parameter is a shade of blue. For our purposes a color is blue if the following is true:

```
The blue component is at least 150 and
the red, when multiplied by 1.4, is less than the blue component and
the green component, when multiplied by 1.4, is less than the blue component
```

Implement method isBlue.

Implement method findBlue so that it does the following:

```
take a picture with the Scribbler
display this image
create a new image with the same dimensions as the
scribbler image
for each pixel p in the new image
     if the corresponding pixel in the scribbler image
is blue
          set the color of p to red
     else
          set the color of p to the same color as the
     original image
display the new image
```

Step 2: Finding arbitrary colors

In the sandbox (or in a new method), create an instance of class Scribbler, connect the object to your robot, and call the `camera` method. The Scribbler continually takes pictures and displays them in a window on your computer.

As you move the mouse around the image window the RGB value of the pixel under the cursor is displayed. Find a bright color in the image, mouse over a pixel on that color and record the RGB color.

There is nothing to submit for this step of the lab.

Step 3: Color distances

One way to determine whether a color is close to another color is to think of the RGB components of a color as a coordinate in 3-dimensional space. It is then relatively easy to calculate the "distance" between these two colors using the algorithm:

```
deltaR = difference between the red components of the colors
deltaG = difference between the green components of the colors
deltaB = difference between the blue components of the colors

distance = Math.sqrt( deltaR*deltaR + deltaG*deltaG +
deltaB*deltaB );
```

The two colors are close if the distance between them is less than some threshold.

Now,
1) Define the constants in method isColor to be the RGB values of one of the colors you found in Step 2 above. Set the value of constant THRESHOLD to 50.

2) Implement a method called calcBlob. It creates an image using the following algorithm:

```
process each pixel p of image
  if isColor returns true when passed the color of p
    set the color of the corresponding pixel in blobImage to white
  else
    set the color of the corresponding pixel in blobImage to black
```

3) Implement a method "blobScribbler" that does the following:

```
Connect to the Scribbler
while the user hasn't pressed a key
  take an image with the Scribbler and display it
  calculate a blob image and display it
close the connection to the Scribbler
```

How well does this program categorize colors based on the reference color you defined? Change the value of THRESHOLD and see how it impacts the categorization of colors.