REFLECTION 30%

1) **Strategies you used to detect the presence and theft how you implemented these strategies in code.**

To begin I used “open_screen_stream();” to turn on the camera.

To detect the theft of the ruby the code I used was the “countrun”. The “countrun” is the number of runs less than a thousand. If the “countrun % 3” was equal to “O” and if the average over time was greater than 2 then the phrase “the Ruby is missing” appears on the screen.
Otherwise a message would appear on screen stating “Ruby is present.” This demonstrated the presence of the ruby.

The way I detected the actual presence of the ruby was by the total amount of redness of the ruby (indicated by the phrase “totred”).
The code I used to accomplish this task was “totred = totred (int)get_pixel(countRow,countCol,0);.”
The initial value of “totred” was 0.
The initial value of “countRow” was 0.
The meaning of “countRow” was the number of rows in the pixel.
The initial value of “countCol” was 0.
The meaning of “countCol” was the number of columns in the pixel.

To determine if the ruby was red enough I used the following code.
In the while loop the “countRow” should have been less than 240 and the “countCol” should have been less than 320. Each time the loop ran the “countRow” and “countCol” should have gradually increased.
For the “get_pixel” the values of “countRow,countCol,0” were the RGB values.

If the “totred” was bigger than 9800000 and the “totred” was less than 11000000 this meant that the average over time was calculated as “averageOverTime-1”.
Otherwise if the “totred” was lesser than 9800000 and more than 11000000 this meant that the average over time was calculated as “averageOverTime+1”.

The “sleep1” was used to indicate how fast the program ran and to determine if the real ruby was replaced with a fake ruby. In this case the code I used was “0,055000” seconds.

The program was meant to keep running until the ruby was missing and then stop. It did this by using “countRun” starting with 0 and gradually increasing to less than 1000. When the ruby went missing it stopped by using the break function in the while loop.
All of this code was also completely effective in preventing the three false alarm attempts (paper, lights and shadow) under the completion section.

To finish I would turn off the camera using the function “close_screen_stream()”

2) Discuss how well your system worked

The tutor said that my system was completely effective in meeting the requirements under the “Core” and “Completion” sections.
However the “Challenge” section my system was only partially effective.
Under the “Challenge” section my system detected when the tutor attempted to replace the ruby with a fake ruby by placing it on top of the original ruby and then removing the original ruby.
My system did not detect when the tutor replaced the original ruby by placing a fake ruby underneath the original ruby and then removing the original. The tutor was able to do this twice and my detection failed both times.

3) Suggest improvements.

I can improve my system by ensuring that the original ruby cannot be replaced by a fake ruby that is brought in from the bottom side of the original ruby. The reason my system failed to detect this was because the tutor replaced the original ruby very fast from the bottom side. Therefore I need to make my system better at recognizing fast replacements that come from the bottom side of the original ruby.
I can do this by altering my code in the following ways.
I must alter my sleep function (“sleep1(0,055000);”).
I must alter the value of my sleep function by decreasing the number to “0,050000”.
I can also alter the range of the numbers of the “totred.” Currently the range is “(totred > 9800000 && totred < 11000000)”.
I could alter this by printing the “totred” values and adjusting it accordingly.
When I do this the altered range of numbers is “totred > 9750000 && totred < 12000000)”.
These improvements should increase my system’s ability to detect fast replacements of the original ruby by the fake ruby coming in from the bottom side of the original ruby.