When computing the yellow value for 100, truncating with "integer" causes the value to go from 255 to 254. Why is this happening?

It's strange, when I save off the value BEFORE calling "integer" and then just return that, it prints "255.000000".

But if I then compute "integer" on that saved-off value of 255.000000, then I get 254.

Make sure you convert your input to a real using (real ...).

Then when you truncate use (int...).

Also, instead of division try to use multiplication. For example (/ input 2) is the same as (* input 0.5)

Scam uses C's printf to print its values, so the behavior you see in Scam is actually C's behavior. You can use Scam's fmt function to see your result with more precision.

In my example, you can see that 1.999999999999 cannot be represented exactly in C's floating point subsystem.

My solution generates 255 for a yellow value of 100, so it is possible to do so.
This issue is coming down to the fact that:

```lisp
scam> (define pi 3.14159265358979323846)
3.1415926536
scam> (sin pi)
1.224647e-16
scam>
```

is not returning 0

Should we be doing something else?

---

Subject: Re: Problem 3
Posted by jarobinson3 on Sun, 21 Aug 2016 23:09:03 GMT

I use the built in sin function and I get the correct answers. Without seeing your code I cannot really make much of a suggestions.

---

Subject: Re: Problem 3
Posted by lusth on Mon, 22 Aug 2016 00:21:39 GMT

You can use a more accurate value of pi if it helps. I just chose the value of pi shown since it worked for my solution (a value of 3.14159 gave me a value of 254 for (yellow 100))

---

Subject: Re: Problem 3
Posted by lusth on Mon, 22 Aug 2016 01:03:31 GMT

Also, cosine wave can be thought of as a left-shifted sine wave.