Ruby trunk - Bug #13116

modulo, divmod range problem: float_val % 1 may return 1.0

01/08/2017 03:42 PM - tompng (tomoya ishida)

Status:	Rejected		
Priority:	Normal		
Assignee:			
Target version:			
ruby -v:	ruby 2.4.0p0 (2016-12-24 revision 57164) [x86_64-darwin16]	**Backport:**	2.2: UNKNOWN, 2.3: UNKNOWN, 2.4: UNKNOWN

Description

mod, divmod range problem: float_val % 1 may return 1.0

x % y sometimes returns y (not in 0 <= result < y)

```
float_val = 0.0.prev_float
float_val % 1 #=> 1.0
Math.sin(2*Math::PI) % 10 #=> 10.0
```

I not sure this is a bug or a feature
but I think returning x.prev_float rather than x would be appropriate.

```
# when would this feature cause problem?:
require 'chunky_png'
image = ChunkyPNG::Image.new 100, 100
points = 100.times.map{|i|[10*Math.sin(Math::PI*i/10), i, 0xaabbccff]}
# render points to an image object(circular boundary)
points.each do |x, y, color|
  image[(x % image.width).floor, (y % image.height).floor] = color
  # => ChunkyPNG::OutOfBounds: Coordinates (100,20) out of bounds!
end
image.save 'out.png'

# to avoid error:
(x % w % w).floor
[(x % w).floor, w - 1].min
xx = (x % w).floor; xx -= 1 if xx == w
# but it's a little bit weird

# sample implementation in ruby:
class Float
  alias_method :original_modulo, :%
  alias_method :original_divmod, :divmod
  def divmod x
    div, mod = original_divmod x
    if mod == x
      mod = x > 0 ? mod.prev_float : mod.next_float
    end
    [div, mod]
  end

  def % x
    mod = original_modulo x
    return mod if mod != x
    x > 0 ? mod.prev_float : mod.next_float
  end
end
```

History

#1 - 01/11/2017 04:51 PM - snood1205 (Eli Sadoff)

For what it's worth, this result fits in with with modulo is documented to be. The documentation states

x.modulo(y) means x-y*(x/y).floor

I implemented this in C to check if the results held and they did. (Note that 0.0.prev_float is -5.0e-324).

```
#include <stdio.h>
#include <math.h>

double modulo(double, double);

int main()
{
        double val = -5.0e-324;
        printf("%.17g\n", modulo(val, 1));
}

double modulo(double x, double y)
{
        return x - y * floor(x / y);
}
```

This returns 1 as the answer as well. The question more so becomes whether or not the ruby implementation of mod should be this or more in line with C's fmod. I would say this is not a bug though because the implementation and the documentation match up.

#2 - 01/11/2017 11:07 PM - nobu (Nobuyoshi Nakada)

- Status changed from Open to Rejected

1.0 - 0.0.prev_float results in 1.0 because of the finite precision.