Document ID: 12_29_97_1
Date Received: 1997-12-29 **Date Revised**: 1998-6-3 **Date Accepted**: 1998-6-13
Curriculum Topic Benchmarks: M2.3.8, M2.4.8, M8.4.1
Grade Level: [9-12] High School
Subject Keywords: modulus, remainder, range, domain, clock, pattern, repetition, periodic
Rating: moderate

Modulus in Real Life

By: Evan M Manning, MS 233-305, Jet Propulsion Lab, 4800 Oak Grove Dr., Pasadena CA 91109 e-mail: evan.m.manning@jpl.nasa.gov

From: The PUMAS Collection http://pumas.jpl.nasa.gov

There is more to modulus than meets the eye. As a computer programmer I use it frequently for a variety of purposes including time and patterns.

Modulus and Division

“Modulus” is the same thing as “remainder.” It is the natural complement of integer division. Many scientific instruments measure time in total seconds. But it is often easier to understand a measurement in minutes and remaining seconds than in total seconds. If we have a period of time measured in seconds, "total_seconds", then we can get minutes, along with the remaining number of seconds, by:

```
  minutes = total_seconds  /  60     [integer division gives only whole numbers]
  seconds = total_seconds mod 60
```

For example, 325 seconds is equal to 5 minutes, 25 seconds.
A similar application of modulus can be used to calculate hours, days, and longer periods of time. Some computers can even calculate both the quotient and the remainder in a single operation.

Range and Domain

From a mathematical perspective, modulus is a particularly handy example of a function with different values for its range and domain. For integer modulus the domain (legal inputs) for the first argument is 0,1,2, ... infinity. The domain for the second argument is 1,2,3, ... infinity. Zero is not part of the domain of the second argument because a number modulus zero would be the remainder of dividing that number by zero. The range (legal outputs) for x modulus y is 0,1,2, ... y-1.

Different Ranges

Of course, in computer programs and other applications, we often want ranges other than 0...y-1. In the case of hours, "total_hours modulus 12" gives an answer in the range 0...11, and might leave us with a time of "Zero O' Clock". We can overcome this with some small adjustments:

```
  hours = ((total_hours - 1) modulus 12) + 1
```

is natural, but gives a negative left-hand argument to modulus when total_hours is 0, so we are better off with:

```
  hours = ((total_hours + 11) modulus 12) + 1
```

Another example is longitude, which is generally restricted to the range -180...180:

```
  restricted_longitude = ((unrestricted_longitude + 180) modulus 360) - 180
```

In both of these cases the right-hand argument of modulus is the "period" of repetition, the final adjustment sets the "floor" of the desired range, and the first adjustment is just the complement of this final adjustment to leave "in-range" inputs unchanged.

Indexing Repetition

Often we use modulus on computers to index a repeating pattern. If I want to make a decorative pattern that looks like:

```
___/\___/\___/\___/\___/\___/\___/\___/\___/\___/\___/\___/\___/\___/\
```

I could store one copy of the pattern in an array:

```
  pattern(0) = "_"
  pattern(1) = "_"
  pattern(2) = "_"
  pattern(3) = "/"
  pattern(4) = "\"
```

And then use modulus to insure that I always index this array with values between 0 and 4:

```
  for raw_index = 0 to 69
      restricted_index = raw_index modulus 5
      if (restricted_index == 0)
          print pattern(0)
      if (restricted_index == 1)
          print pattern(1)
      if (restricted_index == 2)
          print pattern(2)
      if (restricted_index == 3)
          print pattern(3)
      if (restricted_index == 4)
          print pattern(4)
  endfor
```

Or just:

```
  for raw_index = 0 to 69
      restricted_index = raw_index modulus 5
      print pattern(restricted_index)
  endfor
```