99 questions/Solutions/34

From HaskellWiki

< 99 questions | Solutions

(**) Calculate Euler's totient function phi(m).

Euler's so-called totient function phi(m) is defined as the number of positive integers r (1 <= r < m) that are coprime to m.

```
totient 1 = 1
totient a = length $ filter (coprime a) [1..a-1]
 where coprime a b = gcd a b == 1
```

We take coprime from the previous exercise and give it to filter, which applies it to each element of a list from 1 to one less than the number, returning only those that are true. length tells us how many elements are in the resulting list, and thus how many elements are coprime to n

Or slightly more concise, using list comprehension:

```
totient n = length [x | x <- [1..n], coprime x n]
```

For very large numbers, however, the above algorithms become very slow. The Wikipedia article for Euler's totient function (http://en.wikipedia.org /wiki/Euler%27s_totient_function) provides an algorithm that uses the number's prime factors.

```
import Data.List (nub)
import Data.Ratio
totient :: (Integral a) => a -> a
totient 1 = 1
totient n = numerator ratio `div` denominator ratio
 where ratio = foldl (\acc x -> acc * (1 - (1 % x)))
                 (n % 1) $ nub (primeFactors n)
```

This example uses Data.Ratio to ensure no precision is lost. It also relies on a function primeFactors (not shown) that returns a list of all a number's prime factors.

Category:

- Programming exercise spoilers