# Relation Haskell Exercises

Young W. Lim

2018-10-25 Tue

# Outline

# Based on

"The Haskell Road to Logic, Maths, and Programming",
K. Doets and J. V. Eijck

```
module REL

where

import List
import SetOrd
```

```
:load REL
```

# Relation Composition

```
Prelude> :load REL
[1 of 2] Compiling SetOrd           ( SetOrd.hs, interpreted )
[2 of 2] Compiling REL             ( REL.hs, interpreted )
Ok, modules loaded: REL, SetOrd.
*REL> r
{(0,2),(0,3),(1,0),(1,3),(2,0),(2,3)}
*REL> r2
{(0,0),(0,3),(1,2),(1,3),(2,2),(2,3)}
*REL> r3
{(0,2),(0,3),(1,0),(1,3),(2,0),(2,3)}
*REL> r4
{(0,0),(0,3),(1,2),(1,3),(2,2),(2,3)}
*REL>
```

# divisor pairs of n

```
divisors :: Integer -> [{Integer, Integer}]
divisors n = [ (d, quot n d) | d <- [1..k], rem n d == 0]
  where k = floor (sqrt (fromInteger n))
```

```
prime''= Integer -> Bool
prime'' = \n -> divisors n == [(1,n)]
```

# proper divisor

```haskell
divs :: Integer -> [Integer]
divs n = (fst list) ++ reverse (snd list)
  where list = unzip (divisors n)

properDivs :: Integer -> [Integer]
properDivs n = init (divs n)

perfect :: Integer -> Bool
perfect n = sum (properDivs n) == n
```