

Induction Haskell Exercises

Young W. Lim

2018-10-13 Sat

1 Based on

2 Induction and Recursion

- Using REL.hs
- Various Sums of Integers
- Recursion over Integer Numbers

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

I, the copyright holder of this work, hereby publish it under the following licenses: GNU head Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled GNU Free Documentation License.

CC BY SA This file is licensed under the Creative Commons Attribution ShareAlike 3.0 Unported License. In short: you are free to share and make derivative works of the file under the conditions that you appropriately attribute it, and that you distribute it only under a license compatible with this one.

```
module IAR
```

```
where
```

```
import List
```

```
import STA: (display)
```

```
:load REL
```

```
- changes made
```

```
add :: [Natural] -> Natural
```

```
add = foldr plus Z
```

```
mlt :: [Natural] -> Natural
```

```
mlt = foldr mult (S Z)
```

```
rev :: [a] -> [a]
```

```
rev = foldl (\ xs x -> x:xs) []
```

```
rev' :: [a] -> [a]
```

```
rev' = foldr (\ x xs -> xs ++ [x]) []
```

Sum of Odd Integers

- $\sum_{k=1}^n 2k - 1 = n(n + 1) - n = n^2$
- `sumOdds' :: Integer -> Integer`
`sumOdds' n = sum [2*k - 1 | k <- [1..n]]`

```
sumOdds :: Integer -> Integer
sumOdds n = n^2
```

```
*Main> [2*k-1 | k <- [1..10]]
[1,3,5,7,9,11,13,15,17,19]
*Main> [2*k | k <- [1..10]]
[2,4,6,8,10,12,14,16,18,20]
*Main> sumOdds' 10
100
*Main> sumOdds 10
100
```

Sum of Even Integers

- $\sum_{k=1}^n 2k = n(n+1) = n^2 + n$
- `sumEvens' :: Integer -> Integer`
`sumEvens' n = sum [2*k | k <- [1..n]]`

```
sumEvens :: Integer -> Integer
sumEvens n = n * (n+1)
```

```
*Main> [2*k | k <- [1..10]]
[2,4,6,8,10,12,14,16,18,20]
*Main> sumEvens' 10
110
*Main> sumEvens 10
110
```

Sum of Integers

- $\sum_{k=1}^n k = \frac{1}{2}n(n+1)$
- `sumInts' :: Integer -> Integer`
`sumInts' n = sum [1..n]`

```
sumInts :: Integer -> Integer
sumInts n = n * (n+1) / 2
```

```
*Main> [1..10]
[1,2,3,4,5,6,7,8,9,10]
*Main> sumInts' 10
55
*Main> sumInts 10
55
```

Sum of Squares

- $\sum_{k=1}^n k = \frac{1}{6}n(n+1)(2n+1)$
- `sumSquares' :: Integer -> Integer`
`sumSquares' n = sum [k^2 | k <- [1..n]]`

```
sumSquares :: Integer -> Integer
sumSquares n = (n*(n+1)*(2*n+1)) `div` 6
```

```
*Main> [k^2 | k <- [1..10]]
[1,4,9,16,25,36,49,64,81,100]
*Main> sumSquares' 10
385
*Main> sumSquares 10
385
```


Sum of Cubes

- $\sum_{k=1}^n k = \left\{ \frac{1}{2}n(n+1) \right\}^2$
- `sumCubes' :: Integer -> Integer`
`sumCubes' n = sum [k^3 | k <- [1..n]]`

```
sumCubes :: Integer -> Integer
sumCubes n = (n*(n+1) `div` 2)^2
```

```
*Main> [k^3 | k <- [1..10]]
[1,8,27,64,125,216,343,512,729,1000]
*Main> sumCubes' 10
3025
*Main> sumCubes 10
3025
```

- Recursion

```
sum :: [Integer] -> Integer
sum []      = 0
sum (x:xs) = x + sum xs
```

https://en.wikibooks.org/wiki/Haskell/Lists_III

- Iteration

```
import Control.Monad.Trans.State

accumulate :: Int -> State Int Int
accumulate i = do n <- get
                  put (n+i)
                  return n

execState (mapM accumulate [1..10]) 0
```

foldr

- `foldr :: (a -> b -> b) -> b -> [a] -> b`
`foldr f acc [] = acc`
`foldr f acc (x:xs) = f x (foldr f acc xs)`
- `foldr f acc (a:b:c:[]) = f a (f b (f c acc))`

https://en.wikibooks.org/wiki/Haskell/Lists_III

foldl

- $\text{foldl} :: (a \rightarrow b \rightarrow a) \rightarrow a \rightarrow [b] \rightarrow a$
 $\text{foldl } f \text{ acc } [] = \text{acc}$
 $\text{foldl } f \text{ acc } (x:xs) = \text{foldl } f (f \text{ acc } x) xs$
- $\text{foldl } f \text{ acc } (a:b:c:[]) = f (f (f \text{ acc } a) b) c$

https://en.wikibooks.org/wiki/Haskell/Lists_III

foldr and foldl

- `foldl (-) 6 [3, 2, 1] == ((6 - 3) - 2) - 1 -- True`
`foldr (-) 6 [1, 2, 3] == 1 - (2 - (3 - 6)) -- True`
- `GHCi> foldl (-) 6 [3, 2, 1] == 6 - 3 - 2 - 1`
`True`
`GHCi> foldr (-) 6 [1, 2, 3] == 6 - 3 - 2 - 1`
`False`

https://en.wikibooks.org/wiki/Haskell/Lists_III

Recursive definition of Integer Numbers

- `data Natural = Z | S Natural deriving (Eq, Show)`

- using successor S

```
*Main> a1 = S(Z)
```

```
*Main> a2 = S(a1)
```

```
*Main> a3 = S(a2)
```

```
*Main> a4 = S(a3)
```

```
*Main> a1 ..... 1
```

```
S Z
```

```
*Main> a2 ..... 2
```

```
S (S Z)
```

```
*Main> a3 ..... 3
```

```
S (S (S Z))
```

```
*Main> a4 ..... 4
```

```
S (S (S (S Z)))
```

Recursive definition of +

- $m + 0 := m$
- $m + (n + 1) := (m + n) + 1$

- $\text{plus } m \text{ Z} = m$
 $\text{plus } m \text{ (S } n) = \text{S (plus } m \text{ } n)$

$m \text{ 'plus' } Z = m$
 $m \text{ 'plus' } (S \ n) = S \ (m \text{ 'plus' } n)$

- $\text{plus } 2 \text{ Z} = 2$
 $\text{plus } 2 \text{ (S } 3) = \text{S (plus } 2 \text{ } 3) = 6$
 $\text{plus } S \text{ (S } Z) \text{ (S (S (S (S } Z))))} = S \text{ (S (S (S (S } Z))))}$

Recursive definition of $*$

- $m \cdot 0 := 0$
- $m \cdot (n + 1) := (m \cdot n) + m$

- `mult m Z = Z`
`mult m (S n) = plus (mult m n) m`

`m 'mult' Z = Z`
`m 'mult' (S n) = (m 'mult' n) 'plus' m`

- `mult 2 (S 3) = plus (mult 2 3) 2 = 8`
`mult S (S Z) (S (S (S (S Z)))) = S (S (S (S (S (S (S (S Z)))))))`

Recursive definition of exponent

- $m^0 := 1$
- $m^{n+1} := (m^n) \cdot m$

- $\text{expn } m \ Z = (S \ Z)$
 $\text{expn } m \ (S \ n) = \text{mult } (\text{expn } m \ n) \ m$

 $m \ \text{'expn'} \ Z = (S \ Z)$
 $m \ \text{'expn'} \ (S \ n) = (m \ \text{'expn'} \ n) \ \text{'mult'} \ m$

- $\text{expn } 2 \ (S \ 2) = \text{mult } (\text{expn } 2 \ 2) \ 2 = 8$
 $\text{expn } S \ (S \ Z) \ (S \ (S \ (S \ Z))) = S \ (S \ (S \ (S \ (S \ (S \ (S \ (S \ Z)))))))$