Induction Haskell Exercises

Young W. Lim

2018-10-13 Sat

Outline

Based on

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

Using IAR.hs

```
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
  ```
  ```

sum recursive implementations

- Recursion

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

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

sum iterative implementations

- 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

- ```
  foldl :: (a -> b -> a) -> a -> [b] -> a
  foldl f acc []      =  acc
  foldl f acc (x:xs) =  foldl f (f acc x) xs
  ```
- `foldl f acc (a:b:c:[]) = f (f (f 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 defintion 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 defintion of +

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

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

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

- ```
  plus 2 Z = 2
  plus 2 (S 3) = S (plus 2 3) = 6
  plus S (S Z) (S (S (S (S Z)))) = S (S (S (S (S (S Z)))))
  ```

Recursive defintion 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 defintion of exponent

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

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

  m `expn` Z = (S Z)
  m `expn` (S n) = (m `expn` n) `mult` m
  ```

- ```
  expn 2 (S 2) = mult (expn 2 2) 2 = 8
  expn S (S Z) (S (S (S Z))) = S (S (S (S (S (S (S (S Z)))))))
  ```