

# Function Haskell Exercises

Young W. Lim

2018-09-18 Tue

- 1 Based on
- 2 Function
  - Using FCT.hs

## "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.

# Using FCT.hs

```
module FCT
```

```
where
```

```
    :load FCT
```

```
import List
```

# Relation Composition

```
GHCi, version 7.10.3: http://www.haskell.org/ghc/  :? for help
Prelude> :load FCT
[1 of 1] Compiling FCT                ( FCT.hs, interpreted )
Ok, modules loaded: FCT.
*FCT> image (*2) [1,2,3]
[2,4,6]
*FCT> injective (*2) [1, 2, 3]
True
*FCT> injective (^2) [1, 2, 3]
True
*FCT> injective (^2) [-1, -2, -3]
True
*FCT> injective (*2) [-1, -2, -3]
True
*FCT> surjective (*2) [1, 2, 3] [0, 1, 2, 3]
False
*FCT> surjective (*2) [1, 2, 3] [1, 2, 3]
False
*FCT> surjective (*2) [1, 2, 3] [2, 4, 6]
True
```

- $f(x) = x^2 + 1$

```
f x = x^2 + 1
```

- $abs(x) = |x|$

```
absReal x | x >= 0    = x  
          | otherwise = -x |
```

- the identity function

```
id :: a -> a  
ix x = x
```

# From a function to a list

- converting a function to a list

```
list2fct :: Eq a => [(a,b)] -> a -> b
list2fct [] _ = error "function not total"
list2fct ((u,v):uvs) x | x == u    = v
                       | otherwise = list2fct uvs x
```

- examples

```
*Main> fct2list f [1, 2, 3]
[(1,2),(2,5),(3,10)]
*Main> lst = fct2list f [1, 2, 3]
*Main> lst
[(1,2),(2,5),(3,10)]
```

# From a list to a function

- converting a list to a function

```
fct2list :: (a -> b) -> [a] -> [(a,b)]  
fct2list f xs = [ (x, f x) | x <- xs ]
```

- examples

```
*Main> lst  
[(1,2),(2,5),(3,10)]  
*Main> list2fct lst 1  
2  
*Main> list2fct lst 2  
5  
*Main> list2fct lst 3  
10
```