Higher Order Functions

A function is called a "higher-order function"

- If it takes a function as its argument or
- If it returns a function

a. Curried Functions

```
Prelude> let add a b = a + b
Prelude> add 2 3
5
Prelude> let newadd = add 6
Prelude> add 6 3
9
Prelude> newadd 3
9
```

b. map

```
Prelude> map newadd [1,2,3,4,5]
[7,8,9,10,11]
```

c. filter

```
Prelude> filter even [1,2,3,4,5]
[2,4]
```

d. foldr & foldl

```
Prelude> foldr (-) 1 [4,8,5]
0
```

Function (Section) **Base Value** **List**

foldr (-) 1 [4,8,5]
4 – (foldr (-) 1 [8,5])
4 – (8 – (foldr (-) 1 [5]))
4 – (8 – (5 – (foldr (-) 1 [])))
4 – (8 – (5 – 1))
4 – (8 - 4)
4 – 4
0

```
Prelude> foldl (-) 1 [4,8,5]
-16
```

foldl (-) 1 [4,8,5]
foldl (-) (1 - 4) [8,5]
foldl (-) ((1 – 4) - 8) [5]
foldl (-) (((1 - 4) - 8) - 5) []
((1 - 4) - 8) – 5
((-3) - 8) – 5
(-11) – 5
-16

e. zipwith

```
Prelude> zipWith (+) [1,2,3] [2,3,4]
[3,5,7]
Prelude> zipWith max [1,2,3] [2,3,4]
[2,3,4]
```

Practical Exercises:

1. The implementations of the `mapQuestion` and lambdaQuestion2 functions can be found below. You need to trace the following functions and to provide the output of them for the following Haskell function calls.

```
mapQuestion xs = map f xs where f x = x * 2 + 3
lambdaQuestion xs = foldr (\x y -> x + y) 1 xs
```

Function Call	**Output**
`mapQuestion [1,2,3]`	
`lambdaQuestion [1,2,3]`	

2. Implement the set union and the set intersect functions using higher order functions. Sample runs are as follows:

```
setUnion [1,2,3] [3,4]
[1,2,3,4]
```

```
setIntersect [1,2,3] [3,4]
[3]
```

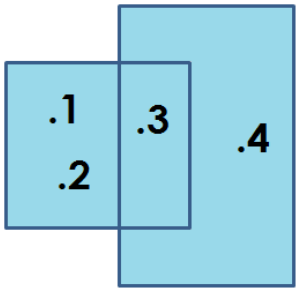

Fig 1. Union Example

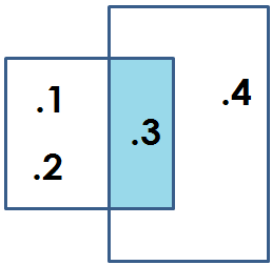

Fig 2. Intersection Example

```
predicate xs x = not (x `elem` xs)
setUnion a b = a ++ filter (predicate a) b
control xs x = x `elem` xs
setIntersect a b = filter (control a) b
```

References:

1. Learn You a Haskell <http://learnyouahaskell.com/chapters>
2. A Gentle Introduction to Haskell <http://www.haskell.org/tutorial/index.html>