Marius Furter
Institut für Mathematik
Universität Zürich

FS 2025

Logic and Foundation with Haskell

Exercise sheet 2

Exercise 1. Implement `map' :: (a -> b) -> [a] -> [b]` using pattern matching.

Exercise 2. Implement `filter' :: (a -> Bool) -> [a] -> [a]` using pattern matching.

Exercise 3. Implement the following functions:

(i) `zip' :: [a] -> [b] -> [(a, b)]`, which takes two list and produces a list of pairs, starting from the left.

(ii) `zipWith' :: (a -> b -> c) -> [a] -> [b] -> [c]`, which takes a function and two lists, and applies the function to each pair of elements in the lists, starting from the left.

(iii) A function `scalarProd :: (Num a) => [a] -> [a] -> a` that returns the scalar product of two lists of numbers.

Exercise 4. Implement `flatten' :: [[a]] -> [a]`, that flattens a list of lists. For example

```
ghci> flatten' [[1,3], [4,5]]
[1,3,4,5]
```

Exercise 5. Implement a function `doubleEveryOther :: [Integer] -> [Integer]`, that doubles every second element of a list of integers, starting from the left. For example,

```
ghci> doubleEveryOther [1..6]
[2,2,6,4,10,6]
```

Exercise 6 (Tricky)**.** Implement a function `toDigits :: Integer -> [Integer]`, that takes an integer and returns a list of its digits. For example, `toDigits 123 = [1,2,3]`. It should return an empty list if the number is zero or negative.