Übungen zu Rekursion - Programmierung und Modellierung 2016

Alexander Isenko

July 17, 2016

Besprechung am 22.Juli 2016

Aufgabe 1

Definieren sie folgende Funktionen:

a) `intersperse :: a -> [a] -> [a]`

Diese Funktion nimmt ein Separator, eine Liste und packt zwischen jedes Element den Seperator. Es kommt kein Separator vor das erste oder hinter das letzte Element der Liste.

```
> intersperse ',' "hallo"
"h,a,l,l,o"

> intersperse 0 [1,2,3]
[1,0,2,0,3]

> intersperse 0 [1]
[1]
```

Lösung:

```haskell
intersperse :: a -> [a] -> [a]
intersperse x []  = []
intersperse x [y] = [y]
intersperse x (y:ys) = y : x : intersperse x ys

-- Z.3: Der Fall ist wichtig, da wir kein 'x'
-- hinter dem letzten Element haben wollen
```

b) `at :: [a] -> Int -> a`

Diese Funktion gibt mir das Element am jeweiligen Index der Liste zurück. Fehlerbehandlung sind nicht nötig. `(!!)` darf nicht benutzt werden.

```
> at [1,2,3] 0
1

> at "hallo" 4
'o'
```

Lösung:

```haskell
at :: [a] -> Int -> a
at (x:_)  0 = x
at (x:xs) n = at xs (n-1)
```

c) `take :: Int -> [a] -> [a]`

Diese Funktion nimmt eine Zahl `n` und eine Liste `xs` und gibt den Prefix der Liste mit Länge `n` zurück oder die Liste selbst, falls `n > length xs`. Fehlerbehandlung sind nicht nötig.

```
> take 3 [1,2,3,4,5]
[1,2,3]

> take 10 [1,2,3]
[1,2,3]

> take 0 [1,2,3]
[]
```

Lösung:

```haskell
take :: Int -> [a] -> [a]
take 0    xs  = []
take n    []  = []
take n (x:xs) = x : take (n-1) xs
```

d) `repeat :: a -> [a]`

Diese Funktion nimmt ein Argument x und erstellt eine unendlich große Liste mit aus x'n

```
-- Pseudobeispiel:
--   repeat 1 => [1,1,1,1,1...

> take 10 (repeat 1)
[1,1,1,1,1,1,1,1,1,1]

> take 5 (repeat 'a')
"aaaaa"

> take 2 (repeat [])
[[],[]]
```

Lösung:

```
repeat :: a -> [a]
repeat x = x : repeat x
```

e) `divBy3 :: [Int] -> [Int]`

Definieren sie mithilfe von List-Comprehentions eine Funktion die alle durch 3 teilbaren Zahlen zurückgibt.

```
> divBy3 [1..10]
[3,6,9]

> divBy3 [(-1),(-2)..(-10)]
[-3,-6,-9]
```

Lösung:

```
divBy3 :: [Int] -> [Int]
divBy3 list = [x | x <- list, mod x 3 == 0]
```