

Übungen zu Funktionaler Programmierung

Übungsblatt 4

Ausgabe: 10.11.2017, **Abgabe:** 17.11.2017 – 16:00 Uhr, **Block:** 2

Aufgabe 4.1 (3 Punkte) *Funktionslifting auf Listen*

Implementieren Sie folgenden Aufgaben mithilfe der Funktion `map` oder `zipWith`. Diese müssen sinnvoll eingesetzt werden.

- a) `double :: [Int] -> [Int]` multipliziert alle Ganzzahlen in einer Liste mit zwei.
Beispiel: `double [1,2,3] ~> [2,4,6]`
- b) `funs :: Int -> [Int]` erzeugt eine Liste mit drei Werten. Dem Doppeltem der Eingabe, dem Nachfolger der Eingabe und die Eingabe hoch 2.
Beispiel: `funs 3 ~> [4,6,9]`
- c) `toUnicode :: String -> [Int]` wandelt einen `String` in eine Liste der Unicode-Codierungen der einzelnen Zeichen um. Mit der Funktion `fromEnum` kann ein Zeichen (`Char`) in seine Codierung gewandelt werden.
Beispiel: `toUnicode "%hello!" ~> [37,104,101,108,108,111,33]`

Aufgabe 4.2 (3 Punkte) *Listenfaltung*

Implementieren Sie folgenden Aufgaben mithilfe der Listenfaltungen `foldl` oder `foldr`. Diese müssen sinnvoll eingesetzt werden.

- a) `catMaybes :: [Maybe a] -> [a]` entfernt alle `Nothing` und gibt nur eine Liste aller `Just`-Werte zurück.
Beispiel: `catMaybes [Just 3, Nothing, Just 5] ~> [3,5]`
- b) `count :: [Color] -> Counter` zählt die Anzahl der Farben in einer Liste. Die Typen `Color` und `Counter` sind wie folgt definiert:

```
data Color = Red | Green | Blue
data Counter = Counter { red, green, blue :: Int } deriving Show
```

Beispiel: `count [Red,Red,Blue] ~> Counter {red = 2, green = 0, blue = 1}`

Aufgabe 4.3 (3 Punkte) *Listenkomprehension*

Definieren Sie folgende Funktionen mithilfe der Listenkomprehension.

- a) `divisors :: Int -> [Int]` gibt eine Liste aller Teiler des Eingabewertes.
Beispiel: `divisors 12 ~> [1,2,3,4,6,12]`
- b) `codes :: [[(Char,Int)]]` gibt alle Lösungen für das Kryptogramm *zwei + vier = sechs*.
- c) `solutions :: [(Int, Int, Int)]` enthält Tripel $(x, y, z) \in \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$, welche die Gleichung $2x^3 + 5y + 2 = z^2$ lösen. Nehmen Sie für x , y und z nur Werte von 0 bis 100.

Aufgabe 4.4 (3 Punkte) *Unendliche Listen*

- a) Definieren Sie eine unendliche Liste `evens`, die alle geraden natürlichen Zahlen aufzählt.
- b) Werten Sie folgenden Haskell-Ausdruck mit *lazy* aus (leftmost-outermost).
`iterate (*2) 3 !! 2`