technische universität dortmund

FLDIT

Prof. Dr. Peter Padawitz

Lehrstuhl 1
Logik in der Informatik

Jos Kusiek (*jos.kusiek@tu-dortmund.de*)
Lukasz Czajka (*lukasz.czajka@tu-dortmund.de*)

Wintersemester 2018/2019

Übungen zu Funktionaler Programmierung
Übungsblatt 4

Ausgabe: 2.11.2018, **Abgabe:** 9.11.2018 – 16:00 Uhr, **Block:** 2

Das Übungsblatt behandelt Themen bis einschließlich Folie 72.

Aufgabe 4.1 (4 Punkte) *Listenfunktionen implementieren*
Implementieren Sie folgende Listenfunktionen in Haskell und geben Sie die Typen der Funktionen an. Es dürfen nur die angegebenen Hilfsfunktionen benutzt werden. Die Typen sollten möglichst allgemein sein.

a) Die Funktion `shift` erhält eine Ganzzahl und eine Liste. Die Zahl gibt an, wie viele Elemente vom Anfang der Liste an das Ende angehangen werden. Die Hilfsfunktionen `(-)` und `(++)` dürfen benutzt werden.
Beispiel: `shift 2 [1,2,3,4,5,6]` ⇝ `[3,4,5,6,1,2]`

b) Die Funktion `removeLetterA` entfernt alle Vorkommen des Großbuchstaben A aus einem `String`.
Beispiel: `removeLetterA "BANANA"` ⇝ `"BNN"`

Aufgabe 4.2 (6 Punkte) *Funktionslifting auf Listen*
Implementieren Sie folgende Aufgaben mithilfe der Funktion `map` oder `zipWith`. Diese müssen sinnvoll eingesetzt werden.

a) `cap :: String -> String` wandelt alle Buchstaben in einem `String` in Großbuchstaben.
Hinweis: Die Funktion `toUpper` wandelt einen einzelnen Buchstaben in einen Großbuchstaben.
Beispiel: `cap "Hello, world"!` ⇝ `"HELLO, WORLD"!`

b) `lesser :: [Int] -> [Int] -> [Int]` vergleicht zwei Listen von Ganzzahlen positionsweise und übernimmt den jeweils kleineren Wert im Ergebnis.
Beispiel: `lesser [5,2,1] [4,2,2]` ⇝ `[4,2,1]`

c) `applyToOne :: [a -> b] -> a -> [b]` wendet alle Funktionen aus einer Liste auf den gleichen Wert an und speichert die einzelnen Ergebnisse in einer Liste.
Beispiel: `applyToOne [(+1),(*2),negate] 2` ⇝ `[3,4,-2]` (= `[2+1,2*2,negate 2]`)

Aufgabe 4.3 (4 Punkte) *Listenfaltung auswerten*
Werten Sie folgende Haskell-Ausdrücke *schrittweise* und *lazy (leftmost-outermost)* aus.

a) `foldl (/) 20 [5,4]`

b) `foldr (/) 20 [5,4]`

Aufgabe 4.4 (4 Punkte) *Listenfaltung implementieren*
Implementieren Sie folgende Aufgaben mithilfe der Listenfaltungen `foldl` oder `foldr`. Diese müssen sinnvoll eingesetzt werden.

a) `countNothing :: [Maybe a] -> Int` zählt alle Vorkommen von `Nothing` in einer Liste.
Beispiel: `countNothing [Nothing, Just 5, Nothing]` $\rightsquigarrow$ `2`

b) `lefts :: [Either a b] -> [a]` entfernt alle `Right`-Werte und gibt eine Liste aller `Left`-Werte aus.
Beispiel: `lefts [Left 3, Right False, Right True, Left 5]` $\rightsquigarrow$ `[3,5]`

Aufgabe 4.5 (6 Punkte) *Listenkomprehension*
Definieren Sie folgende Funktionen mithilfe der Listenkomprehension.

a) `squares :: [(Int,Int)]` Liste jeder geraden Zahl von 0 bis 10 und ihr Quadrat, also `squares == [(0,0),(2,4),(4,16),(6,36),(8,64),(10,100)]`.

b) `solutions :: [(Int, Int, Int)]` enthält Tripel $(x, y, z) \in \mathbb{Z} \times \mathbb{Z} \times \mathbb{Z}$, welche die Gleichung $5x + 3y^2 + 10 == z$ lösen. Nehmen Sie für x, y und z nur Werte von 0 bis 100.

c) `codes :: [[(Char,Int)]]` gibt alle Lösungen für das Kryptogramm $eins + vier = fuenf$.