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 6

Ausgabe: 16.11.2018, **Abgabe:** 23.11.2018 – 16:00 Uhr, **Block:** 2

Aufgabe 6.1 (12 Punkte) *Zahlen als Datentypelemente*
Definieren Sie folgende Konstanten und Datentypen in Haskell mithilfe der in der Vorlesung vorgestellten rekursiven Datentypen `Nat`, `Int'` und `PosNat`.

a) Definieren Sie eine Konstante *drei* = 3 für den Datentyp `Nat`.

b) Definieren Sie eine Konstante *zwei* = 2 für den Datentyp `PosNat`.

c) Definieren Sie eine Konstante *mzwei* = −2 für den Datentyp `Int'`.

d) Erweitern Sie die Datentypen für Zahlen um einen Datentyp `Rat` für rationale Zahlen. Basieren Sie den Datentyp nur auf den Datentypen `Nat`, `Int'` und `PosNat`.

e) Definieren Sie eine Konstante $c = \frac{1}{3}$ für den Datentyp `Rat`.

f) Definieren Sie eine Konstante $c' = -2$ für den Datentyp `Rat`.

Aufgabe 6.2 (12 Punkte) *Rekursive Datentypen*
Definieren Sie folgende Haskell-Funktionen.

a) `natTake :: Nat -> [a] -> [a]`, wie `take` für den Datentyp `Nat`.

b) `natHoch :: (a -> a) -> Nat -> a -> a`, wie `hoch` (Folie 40) für den Datentyp `Nat`.

c) `colistConc :: Colist a -> Colist a -> Colist a`, wie `(++)` für `Colist a`.

d) `colistReverse :: Colist a -> Colist a`, wie `reverse` für `Colist a`.
Hinweis: Es empfiehlt sich die iterative Variante.

e) `stTakeWhile :: (a -> Bool) -> Stream a -> [a]`, wie `takeWhile` für `Stream a`.

f) `stZipWith :: (a -> b -> c) -> Stream a -> Stream b -> Stream c`,
wie `zipWith` für `Stream a`.