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 8

**Ausgabe:** 30.11.2018, **Abgabe:** 7.12.2018 – 16:00 Uhr, **Block:** 3

**Aufgabe 8.1** (6 Punkte) *Typklassen*
Definieren und Instanziieren Sie eine eigene Typklasse.

a) Schreiben Sie eine Klasse für eine überladene Funktion `genDrop`. Diese soll sich wie `drop` verhalten, aber nicht auf den Zahlentyp `Int` beschränkt sein. (2 Punkte)

b) Instanziieren Sie die Klasse für `Int`, `Nat`, `PosNat` und `Int'`. (4 Punkte)

**Aufgabe 8.2** (12 Punkte) *Typklassen*
Instanziieren Sie mehrere Typklassen für den Datentyp `Int'`.

a) Schreiben Sie eine Instanz der Klasse `Enum`. Es ist ausreichend die Funktionen `toEnum` und `fromEnum` zu definieren. (6 Punkte)

Beispiel: `map fromEnum [Minus One .. Plus (Succ' One)]` ⇝ `[-1,0,1,2]`

b) Schreiben Sie Instanzen für die Klassen `Num`, `Eq`, `Ord` und `Show`. Die Klassenfunktionen sollen sich wie für den Typ `Int` verhalten. (4 Punkte)

Beispiel: `show $ Minus $ Succ' One` ⇝ `"-2"`

*Hinweis:* Bereits definierte Funktionen dürfen wiederverwendet werden. Außerdem kann die Funktion `fromIntegral` genutzt werden, um Werte vom Typ `Integer` in den Typ `Int` zu wandeln.

c) Ändern Sie den Typ der Liste `solutions` von Übungsblatt 5 in `[(Int',Int',Int')]` und passen Sie Ihre Lösung entsprechend an. Sie dürfen nur notwendige Änderungen vornehmen. (2 Punkte)

**Aufgabe 8.3** (6 Punkte) *Binäre Bäume*
Definieren Sie folgende Funktionen über binäre Bäume.

a) `sizeBintree :: Bintree a -> Int` – Gibt die Anzahl der Knoten in einem binären Baum wieder.

Beispiel: `sizeBintree btree1` ⇝ `6`

b) `zipBintree :: Bintree a -> Bintree b -> Bintree (a,b)` – Ähnlich wie `zip`, nur auf binären Bäumen. Es werden die Werte mit gleicher Knotenposition zu einem Tupel im Ergebnisbaum zusammengefasst. Gibt es eine Knotenposition nur in einem Baum, entfällt der Knoten im Ergebnis.

Beispiel: `zipBintree btree1 btree4`
⇝ `(6,121)((7,106)((11,99)((55,55),(33,33)),),(9,9))`

c) `getSubbintree :: Bintree a -> Node -> Maybe (Bintree a)` – Gibt den Teilbaum zu einem Knoten aus. Existiert der angegebene Knoten nicht, wird `Nothing` ausgegeben.

Beispiel: `getSubbintree btree1 [Links,Links]` ⇝ `Just 11(55,33)`