

Übungen zu Funktionaler Programmierung

Übungsblatt 10

Ausgabe: 16.12.2016, **Abgabe:** 9.1.2017 - 12:00 Uhr

Aufgabe 10.1 (3 Punkte) Definieren Sie folgende Funktionen als Baumfaltungen (`foldBtree` bzw. `foldTree`) und geben Sie den Typ an. Wählen Sie den Typ möglichst allgemein.

product_ Produkt aller Zahlen in einem Baum mit beliebigen Ausgrad (`Tree`).

inorderB Gibt die Knoten eines Binärbaumes (`Bintree`) als Liste in symmetrischer Reihenfolge (in-order) wieder.

Aufgabe 10.2 (3 Punkte)

1. Schreiben Sie die Faltung `foldNat` für den Typen `Nat`.
2. Schreiben Sie eine Funktion `toInt` die Werte vom Typ `Nat` in entsprechende Werte vom Typ `Int` wandelt. Benutzen Sie dazu die Faltung `foldNat`.

Aufgabe 10.3 (3 Punkte) Geben Sie die Kommandosequenz für die Auswertung `execute (exp2code expr) ([], vars)` an. Zu jedem Kommando soll auch der Stapelinhalt (`Stack`) nach der Ausführung angegeben werden. Dabei sei der Ausdruck `expr` und die Belegungsfunktion `vars` wie folgt definiert:

```
expr :: Exp String
expr = Sum [3 :* Var "x", Con 5]
```

```
vars :: Store String
vars "x" = 2
```

Mit `getResult (execute (exp2code expr) ([], vars))` kann das Ergebnis 11 angezeigt werden. Diese Hilfsfunktion wird für die Bearbeitung der Aufgabe nicht benötigt.

```
getResult :: State x -> Int
getResult = head . fst
```

Aufgabe 10.4 (3 Punkte) Schreiben Sie eine überladene `hash`-Funktion, welche einen Hash vom Typ `Int` erzeugt. Instanzieren Sie die Funktion sinnvoll für die Typen `Nat`, `[a]` und `Tree a`.