

9. Übung

Abgabe in der Woche vom 3. - 7. Juli 2006

Aufgabe 27 (4 Punkte)

Definieren Sie eine Funktion f , die als Argumente zwei einstellige Funktionen g_1 und g_2 , mit $g_1, g_2 : \mathbb{N} \rightarrow \mathbb{N}$, erhält und ein Tupel (h_1, h_2) von zwei Funktionen, mit $h_1, h_2 : \mathbb{N} \rightarrow \mathbb{N}$, zurückgibt, so dass

$$\begin{aligned} h_1(x) &= g_1(x), \text{ falls } x \text{ gerade ist, und } h_1(x) = g_2(x), \text{ sonst und} \\ h_2(x) &= g_2(x), \text{ falls } x \text{ gerade ist, und } h_2(x) = g_1(x), \text{ sonst.} \end{aligned}$$

Aufgabe 28 (5 Punkte)

Definieren Sie eine Funktion, die als erstes Argument eine Liste von Funktionen erhält. Das zweite Argument ist eine Liste von Listen von Werten. Die Funktion soll eine Liste von Listen zurückgeben. Das erste Element dieser Liste sind gerade die Rückgabewerte der ersten Funktion in der Funktionsliste sukzessive angewandt auf die Werte des ersten Elementes der Werteliste (2. Argument). Das zweite Element der Rückgabeliste sind die Rückgabewerte der zweiten Funktion aus der Funktionsliste sukzessive angewandt auf die Werte des zweiten Elementes der Werteliste. Und so weiter. Beispiel:

```
f2 [inc, abs] [[-3,2,1],[5,-1,-7,4]] where inc x = x + 1
>> [[-2,3,2],[5,1,7,4]]
```

Aufgabe 29 (3 + 2 + 6 Punkte)

Gegeben sei folgende Datenstruktur zur Speicherung von Binärbäumen mit positiven Integer-Werten:

```
data Baum = Leer | Knoten Int Baum Baum
          deriving Show
```

1. Schreiben Sie eine Funktion `ist_sortiert`, die für einen solchen Baum entscheidet, ob dieser sortiert ist. D.h. der Wert in der Wurzel eines jeden Teilbaums des gegebenen Baums ist echt größer als alle Werte im linken Teilbaum und echt kleiner als alle Werte im rechten Teilbaum.

2. Schreiben Sie eine zweistellige Funktion `ein fuegen`, die einen neuen Wert in einen Baum sortiert einfügt. Duplikate werden dabei nicht eingefügt.
3. Betten Sie den Typ `Baum` derart in die Klasse `Eq` ein, so dass zwei Bäume genau dann gleich sind, wenn sie die gleichen Werte speichern. Gehen Sie davon aus, dass die Bäume immer sortiert sind. Beispiel:

```
Knoten 1 Leer (Knoten 3 Leer (Knoten 5 Leer Leer)) == Knoten 3 (Knoten 1
  Leer Leer) (Knoten 5 Leer Leer)
>> True
```