WS 2017/2018

Funktionale Programmierung

11. Übungsblatt

Prof. Dr. Margarita Esponda

(Abgabetermin: Mi., den 24.01.2018, um 10:10, Hörsaal)

Ziel: Auseinandersetzung mit Lambda-Kalkül.

1. Aufgabe (3 Punkte)

Schreiben Sie folgende rekursive Funktion als **Lambda**-Ausdruck um:

```
f 0 = 1
f 1 = 1
f n = 3*(f (n-1)) + 1
```

2. Aufgabe (4 Punkte)

Vervollständigen Sie die λ-Funktionen für ganze Werte (positive und negative Zahlen), indem Sie die Subtraktion und die Multiplikation definieren.

3. Aufgabe (3 Punkte)

Ganzzahlige Werte, die mit Zahlen-Tupeln dargestellt werden, haben keine eindeutige Darstellung.

Z.B. Die positive Zahl 3 kann als (8,5) oder (3,0) dargestellt werden.

Definieren Sie einen λ-Ausdruck, der eine beliebige Zahl (a,b) in ein Tupel der Form (n, 0) (positive Zahl) oder (0, m) (negative Zahl) umwandelt.

Beispiel: λz.z 3 5 => λz.z 0 2
λz.z 7 2 => λz.z 5 0

4. Aufgabe (4 Punkte)

Definieren Sie für ganzzahlige Werte (positive und negative Zahlen) **Lambda**-Ausdrücke, die die Vergleichsoperationen (<) und (==) berechnen.

5. Aufgabe (3 Punkte)

Definieren Sie einen λ-Ausdruck, der die Länge von zwei Listen vergleicht (≤) und die Werte 1, 0 oder -1 zurückgibt, je nachdem, ob die erste Liste kleiner, gleich oder größer ist.

6. Aufgabe (6 Punkte)

a) Vervollständigen Sie die in der Vorlesung definierten λ-Ausdrücke für Listen mit einer Funktion, die ein Element in einer Liste sucht und entsprechende Wahrheitswerte zurück gibt.

b) Definieren Sie eine λ-Funktion, die ein Element aus einer Liste löscht bzw. der Semantik folgender Haskell-Funktion entspricht:

```
remove  y  []  =  []
remove  y (x:xs) | y==x      = xs
                 | otherwise = x: remove y xs
```

7. Aufgabe (4 Punkte)

a) Geben Sie für folgende Haskell-Ausdrücke äquivalente Lambda Ausdrücke **in Haskell** an.

```
(2 **)
(f.g.h)
```

b) Programmieren Sie mit sinnvoller Verwendung einer anonymen Funktion und der foldl-Funktion eine Variante der reverse-Funkton für Listen.

8. Aufgabe (4 Punkte)

Die Collatz-Folge, die im Jahr 1937 von Lothar Collatz entdeckt worden ist, wird wie folgt definiert.

Die Collatz-Zahl C_{i+1} mit **i>0** wird wie folgt berechnet:

$$C_{i+1} = \begin{cases} \frac{C_i}{2} & \text{wenn } C_i \text{ gerade} \\ C_i \cdot 3 + 1 & \text{wenn } C_i \text{ ungerade} \end{cases}$$

D.h. die Folge startet mit einer beliebigen Zahl **n**. Wenn **n** gerade ist, ist die Folgezahl gleich $\frac{n}{2}$ und wenn **n** ungerade ist, wird die Folgezahl gleich $(n \cdot 3 + 1)$.

Die Vermutung ist, dass unabhängig davon, mit welcher natürlichen Zahl gestartet wird, die Folge immer mit dem Zahlenzyklus 4, 2, 1 endet. Eine Vermutung, die bis jetzt noch nicht bewiesen worden ist.

Definieren Sie unter Verwendung der **fix** Funktion aus den Vorlesungsfolien und Anonyme Funktionen in Haskell eine rekursive Funktion für die Berechnung der Collatz-Folge.

Anwendungsbeispiel:

(**fix collatz**) 10 => [10, 5, 16, 8, 4, 2, 1]