Programmierung

01. Übungsblatt

Zeitraum: 20. – 24. April 2015

Übung 1

(a) Schreiben Sie in Haskell eine Funktion `square :: [Int] -> [Int]`, die jeden Eintrag einer Eingabeliste quadriert.

(b) Schreiben Sie in Haskell eine Funktion `at :: [Int] -> Int -> Int`, die den n-ten Eintrag einer Liste zurückgibt. Die Liste beginnt mit dem 0-ten Eintrag. Was passiert bei einer leeren Liste?

(c) Schreiben Sie in Haskell eine Funktion `dup :: [Int] -> [Int] -> [Int]`, die eine Liste all jener Elemente zurückgibt, die in jeder der beiden Eingabelisten vorkommen. Falls Sie Hilfsfunktionen verwenden, definieren Sie diese vollständig.

Übung 2 (AGS 12.1)

Programmieren Sie eine Funktion `f` in Haskell, die eine Liste von `Int`–Zahlen als Eingabe nimmt und eine Liste von `Int`–Zahlen mit folgenden Eigenschaften liefert:

- Die Ergebnisliste soll nur Zahlen größer Null enthalten.
- Die Zahlen der Ergebnisliste erscheinen in umgekehrter Reihenfolge bezüglich der Zahlen der Ausgangsliste.

Übung 3 (AGS 12.4)

Gegeben sei eine Liste der Bauart $[l_1, l_2, \ldots, l_n]$ mit $l_1, l_2, \ldots, l_n$ jeweils vom Typ `[Int]`. Es soll von jeder Liste l dieses Typs die Länge der längsten Liste, also $\max\{\text{length}(l_i) \mid 1 \leq i \leq n\}$, als Ergebnis berechnet werden.

(a) Geben Sie ein Beispiel für eine Liste dieses Listentyps an und nennen Sie das zugehörige Ergebnis.

(b) Schreiben Sie in Haskell eine Funktion `max_length :: [[Int]] -> Int`, die diese Aufgabe erfüllt und geben Sie abschließend einen Funktionsaufruf an. Wenn Sie in `max_length` Hilfsfunktionen nutzen, müssen Sie für diese die Typdeklarationen und den Programmcode aufschreiben.

Hilfestellung: Ermitteln sie zunächst aus der Liste von Listen die Liste der Längen dieser Listen.

Übung 4

Betrachten Sie die Liste `s = [1,2,3,...]`, die alle positiven ganzen Zahlen in aufsteigender Reihenfolge enthält.

(a) Geben Sie eine Funktion `gen` an, die die Liste `s` erzeugt.

(b) Stellen Sie die Operationsfolgen für den Aufruf `at gen 2` (dabei ist `at` die Funktion aus Aufgabe 1(b)) bei Benutzung der Aufrufstrategien “call by value” und “call by name” (lazy evaluation) dar.