3. Übung (20.–24 April 2020)

Programmierung

Aufgabe 1 (AGS 12.1.41)
Gegeben ist der Datentyp `data RoseTree = Node Int [RoseTree]` eines Baumes, bei dem jeder Knoten eine beliebige Anzahl an Kindbäumen haben kann (gegeben in einer Liste vom Typ `[RoseTree]`).

(a) Geben Sie die Definition der Funktion `countLeaves :: RoseTree -> Int` an, die ermittelt, wie viele Blattknoten ein gegebener Baum vom Typ `RoseTree` enthält. Ein Blattknoten ist ein Knoten mit einer leeren Liste an Kindbäumen.

(b) Geben Sie die Definition der Funktion `evenNodes :: RoseTree -> Bool` an, die zu einem gegebenen Baum vom Typ `RoseTree` ermittelt, ob jeder Knoten eine gerade Anzahl an Nachfolgern hat und in diesem Fall `True` zurückgibt (ansonsten `False`). Sie dürfen dabei auf die Funktion `length :: [RoseTree] -> Int` zurückgreifen, die die Länge einer Liste von Bäumen ermittelt.
Hinweis: In Haskell ist die Funktion ***`length`*** *bereits implementiert, Sie können sie ohne eigene Definition verwenden.*

Aufgabe 2 (12.1.54)
In der Vorlesung wurden die Higher-Order-Funktionen

- `map :: (Int -> Int) -> [Int] -> [Int]`,
- `filter :: (Int -> Bool) -> [Int] -> [Int]`, und
- `foldr :: (Int -> Int -> Int) -> Int -> [Int] -> Int`

vorgestellt. Implementieren Sie eine Funktion `f :: [Int] -> Int` mithilfe von `map`, `filter` und `foldr`, die das Produkt der Quadrate der geraden Zahlen in der Eingabeliste berechnet.
Hinweis: In Haskell sind die Funktionen ***`map`****,* ***`filter`*** *und* ***`foldr`*** *bereits implementiert, Sie können sie ohne eigene Definition verwenden.*

Aufgabe 3 (AGS 12.1.57)
Geben Sie eine Funktion `foldleft :: (Int -> Int -> Int) -> Int -> [Int] -> Int` an, so dass für jedes `f :: Int -> Int -> Int` und $\texttt{a}_0$ `:: Int`, $\texttt{b}_1$, ..., $\texttt{b}_\texttt{k}$ `:: Int`, $\texttt{k} \in \mathbb{N}$ gilt, dass

$$\texttt{foldleft f a}_0\ [\texttt{b}_1,\ \ldots,\ \texttt{b}_\texttt{k}] \;=\; \texttt{f}\ (\texttt{f}\ \cdots\ (\texttt{f a}_0\ \texttt{b}_1)\ \cdots\ \texttt{b}_{\texttt{k}-1})\ \texttt{b}_\texttt{k}\,,$$

also z.B.

$$\texttt{foldleft}\ (+)\ 5\ [1,\ 4,\ 3] \;=\; (+)\ ((+)\ ((+)\ 5\ 1)\ 4)\ 3 \;=\; ((5\ +\ 1)\ +\ 4)\ +\ 3\,.$$

Insbesondere soll `foldleft f a [] = a` gelten.

Zusatzaufgabe 1 (AGS 12.1.11)

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 $\max\{\text{length}(l_i) \mid 1 \leq i \leq n\}$, 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 `maxLength :: [[Int]] -> Int`, die diese Aufgabe erfüllt und werten Sie abschließend einen Funktionsaufruf aus.