Compilerbau

DHBW Campus Horb **Übungsblatt 1**

1 Listen

a) Schreiben sie eine Funktion `lastElement`, welche das letzte Element einer Liste zurück gibt:

```
lastElement :: [a] -> a
```

b) Schreiben sie eine Funktion `stripList`, welche das erste und das letzte Element einer Liste entfernt:

```
stripList :: [a] -> [a]
--Beispiel
stripList "hallo" -- "all"
```

c) Schreiben sie eine Funktion `isPalindrom`, die Palindrome erkennt:

```
isPalindrom :: (Eq a) => [a] -> Bool
--Beispiele
isPalindrom [1, 2, 3] -- False
isPalindrom "anna" -- True
```

2 Arithmetik

a) Schreiben sie eine Funktion `isPrime`, welche ausgibt ob eine Zahle eine Primzahl ist:

```
isPrime :: Num -> Bool
--Beispiel
isPrime 7 -- True
```

b) Schreiben sie eine Funktion `getPrimes`, welche eine Liste aller Primzahlen ausgibt:

```
getPrimes :: [Num]
--Beispiel
head getPrimes -- 2
```

3 curry3, uncurry3

Schreiben Sie die Funktionen

```
curry3 :: ((a, b, c) -> d) -> (a -> b -> c -> d)
uncurry3 :: (a -> b -> c -> d) -> ((a, b, c) -> d)
```

die die jeweiligen Funktionen mit dem Tupel Typ–Konstruktor in äquivalente Funktionen mit dem `->` Typ–Konstruktor umwandeln und umgekehrt.

4 Die Funktion map

Schreiben Sie eine Funktion, `toListofLists :: [a] -> [[a]]`, die jedes Element einer Liste zu einer einelementigen Liste macht. Nutzen Sie dazu die Funktion `map`.

5 Die Funktionen `foldl` und `foldr`

Programmieren Sie eine Funktion `listStringConcat :: [String] -> String`, die alle Elemente einer `String`–Liste aneinander hängt.

Nutzen Sie dazu entweder die Funktion `foldl` oder `foldr`.

Erhalten Sie unterschiedliche Ergebnisse?

6 Abstrakter Datentyp `BBaum`

a) Programmieren Sie den polymorphen abstrakten Datentyp `BBaum a`. `BBaum a` steht für einen Binärbaum und soll zwei Konstruktoren enthalten. Einen Konstruktor für einen leeren Binärbaum und einen Konstruktor der zwei Teilbäume und ein Element zu einem neuen Binärbaum zusammenfügt.

b) Schreiben Sie eine Funktion `mapTree :: BBaum a -> (a -> b) -> BBaum b`. Der Funktionsaufruf `mapTree b f` soll auf jedes Element des Eingabebaums `b` die Funktion `f` anwenden.

c) Schreiben sie ein Funktion `flatBBaum`. Die Funktion konvertiert einen Binärbaum in eine Liste.

```
flatBBaum :: BBaum a -> [a]
```