technische universität dortmund

FLDIT

Lehrstuhl 1
Logik in der Informatik

Prof. Dr. Peter Padawitz

Jos Kusiek (*jos.kusiek@tu-dortmund.de*) Wintersemester 2017/2018

Übungen zu Funktionaler Programmierung
Übungsblatt 3

Ausgabe: 3.11.2017, **Abgabe:** 10.11.2017 – 16:00 Uhr, **Block:** 2

Aufgabe 3.1 (4 Punkte) *Endrekursion*
Formen Sie folgende Funktionen, die Schleifen enthalten, in *endrekursive* Funktionen um.

a) Eine Potenzfunktion, die mit einer Multiplikation arbeitet.

```
int power(int base, int expo) {
    int state = 1;
    while (expo > 0) {
        state = state * base;
        expo = expo - 1;
    }
    return state;
}
```

b) Eine Funktion die alle Zahlen in einem Feld summiert. Benutzen Sie in Haskell eine Liste anstelle des Feldes.

```
int summe(int[] ls) {
    int state = 0;
    int i = 0;
    while (i < ls.length) {
        state = state + ls[i];
        i = i + 1;
    }
    return state;
}
```

Aufgabe 3.2 (4 Punkte) *Listenfunktionen auswerten*
Werten Sie folgende Haskell-Ausdrücke schrittweise aus.

a) `take 2 $ tail [2,3,5,4,1]`

b) `head $ drop 2 [1,4,5,3,2]`

c) `foldl (-) 8 [5, 2]`

d) `foldr (-) 8 [5, 2]`

Aufgabe 3.3 (4 Punkte) *Listenfunktionen implementieren*
Implementieren Sie folgende Listenfunktionen in Haskell und geben Sie die Typen der Funktionen an. Es dürfen keine Hilfsfunktionen benutzt werden. Die Typen sollten möglichst allgemein sein.

a) Die Funktionen `safeHead` und `safeTail` sollen als absturzsichere Versionen der Funktionen `head` und `tail` implementiert werden. Machen Sie gebrauch vom `Maybe`-Datentyp.

b) Die Funktion `listEven` soll prüfen, ob die Anzahl der Elemente in einer Liste gerade sind.