technische universität dortmund

FLDIT

Lehrstuhl 1
Logik in der Informatik

Prof. Dr. Peter Padawitz

Jos Kusiek (*jos.kusiek@tu-dortmund.de*) Wintersemester 2016/2017

Übungen zu Funktionaler Programmierung
Übungsblatt 12

Ausgabe: 20.1.2016, **Abgabe:** 27.1.2017 - 12:00 Uhr

Aufgabe 12.1 (2 Punkte) Gegeben sei folgende Listenkomprehension:

```
solutions :: [(Int , Int , Int )]
solutions = [ (x,y,z) | z <- [0..] , y <- [0..z] , x <- [0..z]
            , 3*x^2 + 2*y + 1 == z]
```

1. Überführen Sie die Listenkomprehension in die do-Notation.
2. Überführen Sie die do-Notation in monadische Operatoren und Funktionen(»=,»,`return`).

Aufgabe 12.2 (2 Punkte) *Hinweis: Für diese Aufgabe muss Abschnitt 7.1* Maybe- und Listenmonaden *gelesen werden.*
Definieren Sie die Haskell-Funktion `f` vom Typ `Int -> Int -> Int -> Maybe Int`. Wie in Aufgabe 11.3 soll diese mit den Funktionen `safeDiv` und `safeSqrt` gelöst werden und die Gleichung $f(x,y,z) = \frac{\sqrt{x}}{\sqrt{\frac{y}{z}}}$ erfüllen. Diesmal soll jedoch sonst nur die Monadeneigenschafft von `Maybe` genutzt werden.

Aufgabe 12.3 (4 Punkte) *Hinweis: Für diese Aufgabe muss Abschnitt 7.10* Schreibermonaden *gelesen werden.*
Schreiben Sie einen Algorithmus, welche die Operation $2*(3+1)+5$ ausführt und dabei jeden Schritt protokolliert.
Gehen Sie dazu schrittweise vor.

1. Schreiben Sie die Funktionen `addW` und `multW`, welche zwei Ganzzahlen addieren bzw. multiplizieren und den Vorgang protokollieren. Definieren Sie die Funktionen mithilfe der do-Notation und der Funktion `tell`. Die Funktion `tell` schreibt einen beliebigen `String` in das Protokoll.
   Vorgaben:

   ```
   type Writer s a = (s,a)

   tell :: s -> Writer s ()
   tell s = (s,())

   addW, multW :: Int -> Int -> Writer String Int
   ```

2. Definieren Sie das Programm `progW` vom Typ `Writer String Int`, welches die Operation $2*(3+1)+5$ ausführt. Nutzen Sie auch hier die do-Notation.
   Beispiel: `progW` $\rightsquigarrow$ `("3+1 = 4\n2*4 = 8\n8+5 = 13\n",13)`

Aufgabe 12.4 (4 Punkte) *Hinweis: Für diese Aufgabe muss Abschnitt 7.9* Lesermonaden *gelesen werden.*
Schreiben Sie die Funktion `bexp2store` aus Aufgabe 7.4 so um, dass sie Gebrauch von der Lesermonade macht. Es gelten folgende Typen:

```
type BStore x = x -> Bool
bexp2store :: BExp x -> Store x -> BStore x -> Bool
```