Westfälische Hochschule

Fachbereich Informatik und Kommunikation
Prof. Dr. Marcel Luis

7. Juni 2016

Aufgabenblatt 8 zu Funktionale Programmierung

Aufgabe 8.1 (Übung)

In der Vorlesung haben wir die Exc-Monade definiert:

```
instance Monad Exc where
  -- return :: a -> Exc a
  return = Return
  -- (>>=) :: Exc a -> (a -> Exc b) -> Exc b
  (Raise e)  >>= q = Raise e
  (Return x) >>= q = q x
```

Beweisen Sie, dass für diese Monade die drei Monadengesetze gelten.

$$\begin{aligned} p >>= return &= p \\ return\ e >>= q &= q\ e \\ (p >>= q) >>= r &= p >>= (\backslash x -> q\ x >>= r) \end{aligned}$$

Aufgabe 8.2

Im Anwendungsbeispiel für den Datentyp `St Int` wird gezählt, wieviele Divisionen bei der Auswertung eines Terms durchgeführt werden. Das korrekte Zählen ergibt sich durch das Erhöhen um 1 innerhalb von `evalSt` und den Startwert 0 für den Zustand innerhalb der Funktion `show`.

Wenn es nur um das Zählen geht (und nicht um „kompliziertere“ Zustandsänderungen), ist die State-Monade nicht notwendig. Man kann das einfacher durch ein Verfahren erreichen, das der Output- bzw. Writer-Monade ähnlich ist.

Realisieren Sie einen Datentyp $Sum\ a$ und eine Funktion $evalSum :: Term \rightarrow Sum\ Int$, sodass bei Auswertung eines Terms der Wert und die Anzahl der Divisionen ermittelt wird.