Proseminar – Vortrag: Monad Transformers

Übungsaufgaben

Wiederholung Monaden:

1.) Beschreiben Sie in Worten, was die Funktion mapTree bewirkt:

```
data Tree a = Leaf a | Branch (Tree a) (Tree a)

mapTree :: (a -> b) -> Tree a -> Tree b
mapTree f (Leaf a) = Leaf (f a)
mapTree f (Branch lhs rhs) = Branch (mapTree f lhs) (mapTree f rhs)
```

2.) Beweisen Sie die Analogie von *Monad Law 1*(return a >>= f ≡ f a) mit der *bindState*-Funktion.
(Hinweis: *Substitution*! Beginne mit returnState a `bindState` f)

```
bindState :: State st a -> (a -> State st b) -> State st b
bindState m k = \st ->
      let (st', a) = m st
          m'       = k a
      in  m' st'

returnState :: a -> State st a
returnState a = \st -> (st, a)
```

Monad Transformers:

3.) Schreiben Sie für die StateMonadTransformer die Funktionen *getT* und *putT*, die zu *getState* und *putState* analog sind.

4.) Überlegen Sie ob man mehr als 2 Monaden mithilfe von Transformers kombinieren kann und was für Eigenschaften sich ergeben würden.