9. Aufgabenblatt zu Funktionale Programmierung vom 02.12.2008.
Fällig: Di, 09.12.2008 / Di, 16.12.2008 (jeweils 15:00 Uhr)

Themen: *Programmieren mit Monaden*

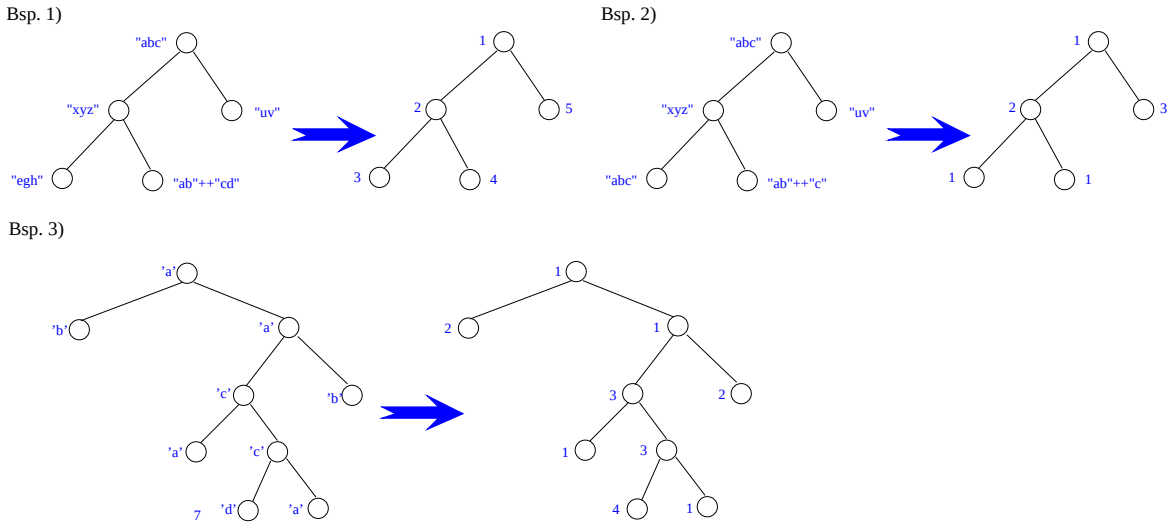
Für dieses Aufgabenblatt sollen Sie Haskell-Rechenvorschriften für die Lösung der unten angegebenen Aufgabenstellungen entwickeln und für die Abgabe in einer Datei namens `Aufgabe9.lhs` ablegen. Sie sollen für die Lösung dieses Aufgabenblatts also wieder ein "literate" Haskell-Skript schreiben. Versehen Sie wieder wie auf den bisherigen Aufgabenblättern alle Funktionen, die Sie zur Lösung brauchen, mit ihren Typdeklarationen und kommentieren Sie Ihre Programme aussagekräftig. Benutzen Sie, wo sinnvoll, Hilfsfunktionen und Konstanten.

Im einzelnen sollen Sie die im folgenden beschriebenen Problemstellungen bearbeiten.

1. Gegeben sei:

   ```
   data Tree a = Leaf a | Node a (Tree a) (Tree a) deriving (Eq,Show)
   ```

   Gesucht ist eine Haskell-Rechenvorschrift `renameTree` mit der Signatur `renameTree :: (Eq a, Show a) => Tree a -> Tree Integer`, die angewendet auf einen Binärbaum einen Binärbaum über ganzen Zahlen zurückliefert. Gleiche Benennungen im Argumentbaum sollen dabei auf gleiche Zahlen im Resultatbaum abgebildet werden. Die Benennungen im Resultatbaum werden den Benennungen im Argumentbaum dabei beginnend mit 1 in aufsteigender Folge entsprechend eines Durchlaufs des Argumentbaums in Präfixordnung vergeben. Die folgende Abbildung illustriert das gewünschte Abbildungsverhalten:



   Hinweis: In Präfixordung wird zunächst die Wurzel, dann der linke Teilbaum, schließlich der rechte Teilbaum eines Baums besucht. Um die schon den im Argumentbaum aufgefundenen Benennungen zugeordneten Zahlen zu verwalten und wiederaufzufinden, bietet sich an, diese Benennungen in einer Tafel (repräsentiert durch ein Liste) abzulegen. Die Indizes dieser Liste können dann direkt oder indirekt die einer Benennung des Argumentbaums zugehörige Zahl angeben. Denken Sie daran, dass der erste Index einer Liste die 0 ist:

   ```
   type Table a = [a]
   ```

2. In dieser Aufgabe betrachten wir die gleiche Aufgabenstellung wie in der vorherigen Teilaufgabe, wollen diese nun aber mittels monadischer Programmierung lösen. Dazu betrachten wir die sog. *Zustandsmonade*, d.h. den Datentyp `State` mit

   ```
   data State s a = St (s -> (a,s))
   ```

   und der Instanzbildung

```
instance Monad (State s) where
  return x     = (St (\s -> (x,s))
  (St m) >>= f =  St (\s -> let (x,s1) = m s
                                St g = f x
                            in g s1)
```

Mithilfe dieses Datentyps lassen sich *Zustandstransformationen* ausdrücken, d.h. Funktionen `f` vom Typ `f :: s -> (a,s)`, die einen initialen Zustand `s1` vom Typ `s` auf ein Paar bestehend aus einem (möglicherweise veränderten) Zustand `s2` vom Typ `s` und einem Wert `a1` vom Typ `a` abbilden.

Gesucht ist nun eine Haskell-Rechenvorschrift `monRenameTree` mit der Signatur `monRenameTree :: (Eq a, Show a) => Tree a -> State (Table a) (Tree Integer)`, die einen Binärbaum wie in der vorigen Teilaufgabe beschrieben in einen Resultatbaum überführt, wobei `Table a` wie in der vorigen Teilaufgabe vorgeschlagen ist, d.h.:

```
type Table a = [a]
```

und zur Buchführung über die Zuordnung von Argumentbaum- zu Resultatbaumbenennungen verwendet wird. Sehen Sie in Ihrer Lösung zusätzlich eine Funktion `mRT` vor mit

```
mRT :: (Eq a, Show a) => Tree a -> Tree Integer
mRT = extract . monRenameTree
```

wobei `extract` eine Extraktionsfunktion mit folgender Signatur ist:

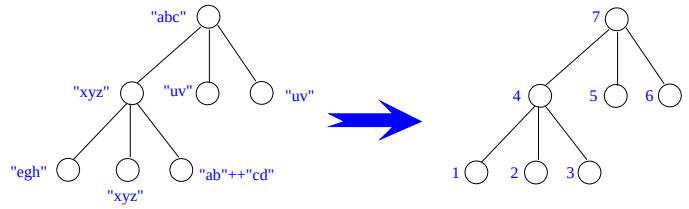
```
extract :: (Eq a, Show a) => (State (Table s) a) -> a}
```

3. Gegeben sei:

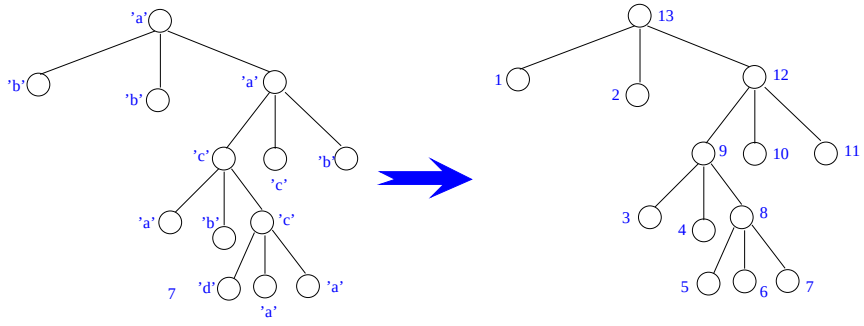
```
data TripleTree a
          = Leaf3 a
            | Node3 a (TripleTree a) (TripleTree a) (TripleTree a) deriving (Eq,Show)
```

Gesucht ist eine Haskell-Rechenvorschrift `numberTree` mit der Signatur `numberTree :: (Eq a, Show a) => TripleTree a -> TripleTree Integer`, die angewendet auf einen Baum einen Baum zurückliefert, in dem alle Knoten und Blätter fortlaufend mit natürlichen Zahlen beginnend mit 1 benannt sind. Der Baum soll dabei in folgender Ordnung durchlaufen werden: Zunächst der linke Teilbaum, dann der mittlere Teilbaum, danach der rechte Teilbaum und schließlich die Wurzel. Die folgende Abbildung illustriert das gewünschte Abbildungsverhalten:

Bsp. 1)



Bsp. 2)



Lösen Sie diese Aufgabe unter Verwendung von (Zustands-) Monaden.

4. Lösen Sie die Aufgabe aus der vorigen Teilaufnahme ohne Verwendung von (Zustands-) Monaden. Die Funkion `numberTree` soll dabei `numberTreeMF` heißen, wobei `MF` an "monadenfrei" erinnert. Hängen Sie das Postfix `MF` auch an alle anderen Namen an, für die es sonst einen Namenskonflikt gäbe.

Hinweis:

- Verwenden Sie *keine* Module. Wenn Sie Funktionen wiederverwenden möchten, kopieren Sie diese in die Abgabedatei `Aufgabe9.hs`. Andere als diese Datei werden vom Abgabeskript ignoriert. Denken Sie daran, dass Sie dieses Mal ein literate Haskell-Skript schreiben sollen.

Frohe Weihnachten

und

einen guten Rutsch ins neue Jahr!