7. Aufgabenblatt zu Funktionale Programmierung vom 30.11.2016.
Fällig: 07.12.2016 (15:00 Uhr)

Themen: *Aufgaben auf Suchbäumen und Multimengen*

Für dieses Aufgabenblatt sollen Sie Haskell-Rechenvorschriften für die Lösung der unten angegebenen Aufgabenstellungen entwickeln und für die Abgabe in einer Datei namens `Aufgabe7.hs` ablegen. Sie sollen für die Lösung dieses Aufgabenblatts also ein "gewöhnliches" 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.

Zur Frist der Zweitabgabe: Siehe allgemeine Hinweise auf Aufgabenblatt 1.

Wir betrachten in dieser Aufgabe folgende Erweiterung zur Darstellung binärer Suchbäume von Aufgabenblatt 4, wobei sich die auf Blatt 4 eingeführten Ordnungskriterien für Bäume in natürlicher Weise auf die Werte des erweiterten Baumtyps übertragen:

```
data Tree a = Nil | Node a Int (Tree a) (Tree a) deriving (Eq,Ord,Show)
```

Werte des erweiterten Suchbaumtyps können zur Darstellung von *Multimengen* verwendet werden, wobei der Zahlwert an einem Knoten die Anzahl der `a`-Werte in der vom Suchbaum repräsentierten Multimenge angibt.

```
type Multiset a = Tree a
```

Zusätzlich führen wir folgende Aufzählungstypen ein:

```
data ThreeValuedBool = TT | FF | Invalid deriving (Eq,Show)
data Order           = Up | Down deriving (Eq,Show)
```

Ein Wert `t` vom Typ `Tree a` *repräsentiert eine Multimenge*, wenn `t` ein Suchbaum (im Sinne von Aufgabenblatt 4) und der Zahlwert an jedem Knoten größer oder gleich 0 ist. Ein Wert `t` vom Typ `Tree a` ist *kanonische Darstellung einer Multimenge*, wenn `t` eine Multimenge repräsentiert und der Zahlwert an jedem Knoten echt größer als 0 ist. Mit diesen Festlegungen ist der Baumwert `Nil` die kanonische Darstellung der leeren Multimenge.

Schreiben Sie Haskell-Rechenvorschriften für folgende Aufgaben:

1. `isMultiset :: Ord a => Tree a -> Bool`
   Angewendet auf einen Baumwert `t`, liefert die Wahrheitswertfunktion `isMultiset` den Wert `True`, wenn `t` Darstellung einer Multimenge ist, sonst `False`.

2. `isCanonicalMultiset :: Ord a => Tree a -> Bool`
   Angewendet auf einen Baumwert `t`, liefert die Wahrheitswertfunktion `isMultiset` den Wert `True`, wenn `t` kanonische Darstellung einer Multimenge ist, sonst `False`.

3. `mkMultiset :: (Ord a,Show a) => Tree a -> Multiset a`
   Angewendet auf einen Baumwert `t`, der ein Suchbaum sein kann oder nicht, liefert die Funktion `mkMultiset` einen Suchbaum `t'` zurück, wobei `t'` alle in `t` vorkommenden `a`-Werte enthält und der zu jedem `a`-Wert A in `t'` gehörige Zahlwert das Maximum aus 0 und der Summe der Zahlwerte der mit A in `t` benannten Knoten ist.

4. `mkCanonicalMultiset :: (Ord a,Show a) => Tree a -> Multiset a`
   Die Funktion `mkCanonicalMultiset` verhält sich wie die Funktion `mkMultiset`, jedoch mit kanonischem Resultat.

5. `flatten :: (Ord a,Show a) => Order -> Multiset a -> [(a,Int)]`
   Angewendet auf eine Multimenge m (gleich ob kanonisch oder nicht), liefert die Funktion `flatten` eine aufsteigend (absteigend) geordnete Liste der in m mindestens mit einem Vorkommen enthaltenen `a`-Werte zusammen mit der Zahl $n > 0$ dieser Vorkommen, wenn `flatten` mit `Up` (`Down`) als erstem Argument aufgerufen wird. Elemente einer nichkanonischen Multimenge mit Anzahl 0 der Vorkommen werden in der Ausgabe von `flatten` also unterdrückt. Ist m keine Multimenge, so liefert `flatten` die leere Liste als Resultat.

6. `isElement :: Ord a => a -> Multiset a -> Int`
   Angewendet auf einen `a`-Wert A und eine Multimenge m (gleich ob kanonisch oder nicht), liefert die Funktion `isElement` die Anzahl der Vorkommen von A in m. Ist m keine Multimenge, so liefert `isElement` das Resultat -1.

7. `isSubset :: Ord a => Multiset a -> Multiset a -> ThreeValuedBool`
   Angewendet auf zwei Multimengen m_1 und m_2 (gleich ob kanonisch oder nicht) liefert die Funktion `isSubset` den Wert `TT`, wenn m_1 eine Teilmenge von m_2 ist, d.h., wenn jeder in m_1 vorkommende a-Wert gleich oft oder öfter auch in m_2 vorkommt, sonst `FF`. Ist eines der Argumente keine Darstellung einer Multimenge, so liefert die Funktion den Wert `Invalid`.

8. `join :: (Ord a,Show a) => Multiset a -> Multiset a -> Multiset a`
   Angewendet auf zwei Multimengen m_1 und m_2 (gleich ob kanonisch oder nicht), liefert die Funktion `join` eine kanonische Multimenge m_3, die die Vereinigung von m_1 und m_2 darstellt. Dabei enthält m_3 jedes Element aus m_1 und m_2 so oft, wie es der Summe der Vorkommen dieses Elements in m_1 und m_2 entspricht. Ist eines der Argumente keine Darstellung einer Multimenge, so liefert die Funktion den Wert `Nil`.

9. `meet :: (Ord a,Show a) => Multiset a -> Multiset a -> Multiset a`
   Analog zu `join`, jedoch liefert `meet` angewendet auf zwei Multimengen m_1 und m_2 (gleich ob kanonisch oder nicht) die kanonische Darstellung m_3 des Durchschnitts von m_1 und m_2. Ist eines der Argumente keine Darstellung einer Multimenge, so liefert die Funktion den Wert `Nil`.

10. `subtract :: (Ord a,Show a) => Multiset a -> Multiset a -> Multiset a`
    Analog zu `join` und `meet`, jedoch liefert `subtract` angewendet auf zwei Multimengen m_1 und m_2 (gleich ob kanonisch oder nicht) die kanonische Darstellung m_3 der Differenz von m_1 und m_2. m_3 enthält jeden in m_1 vorkommenden `a`-Wert so oft, wie er in m_1 vorkommt abzüglich der Zahl der Vorkommen dieses Werts in m_2. Ist diese Differenz kleiner oder gleich 0, ist der entsprechende `a`-Wert nicht in m_3 enthalten. Ist eines der Argumente keine Darstellung einer Multimenge, so liefert die Funktion den Wert `Nil`.

Wichtig: Wenn Sie einzelne Rechenvorschriften aus früheren Lösungen für dieses oder spätere Aufgabenblätter wieder verwenden möchten, so kopieren Sie diese in die neue Abgabedatei ein. Ein `import` schlägt für die Auswertung durch das Abgabeskript fehl, weil Ihre frühere Lösung, aus der importiert wird, nicht mit abgesammelt wird. Deshalb: Kopieren statt importieren zur Wiederwendung!

Haskell Live

Der nächste *Haskell Live*-Termin ist am Freitag, den 02.12.2016.