Programmierung

03. Übungsblatt

Zeitraum: 04. – 08. Mai 2015

Übungsverlegung zum Dies Academicus (06. Mai):
Ü 1. DS und 5. DS finden statt am 13. Mai, SCH/B247, 4. DS

Übung 1 (AGS 12.1.10)
Folgende Definition sei gegeben:

- Ein A-Baum ist entweder leer oder seine Wurzel ist mit A beschriftet und der linke Nachfolger von A ist ein A-Baum und der rechte Nachfolger von A ist ein B-Baum.
- Ein B-Baum ist entweder leer oder seine Wurzel ist mit B beschriftet und der linke Nachfolger von B ist ein B-Baum und der rechte Nachfolger von B ist ein A-Baum.

(a) Geben Sie algebraische Datentypen `TA` bzw. `TB` für A-Bäume bzw. B-Bäume an.
(b) Geben Sie eine Funktion `zahlB` an, die die Anzahl der Vorkommen von B-Knoten in einem A-Baum ermittelt.

Übung 2 (AGS 12.1.23)
Gegeben sei der Typ

```
data Tree = Node Int Tree Tree | NIL
```

(a) Geben Sie eine Funktion `insert :: Tree -> [Int] -> Tree` an, die alle Werte einer Liste von Integer-Zahlen in einen bereits bestehenden *Suchbaum* des o. g. Typs so einfügt, dass die Suchbaumeigenschaft erhalten bleibt. Die Werte der Liste seien paarweise verschieden.
(b) Geben Sie eine Haskell-Funktion einschließlich der Typ-Definiton an, die testet, ob zwei Binärbäume des o. g. Typs identisch sind.

Übung 3 (AGS 12.1.30)
Gegeben sei der folgende algebraische Datentyp für einen knotenbewerteten Binärbaum:

```
data Tree = Branch Int Tree Tree | Nil
```

Schreiben Sie eine Haskell-Funktion `isHeap`, die für jeden gegebenen Baum testet, ob er die Heap-Eigenschaft hat. Ein Baum hat die Heap-Eigenschaft, wenn für jeden Teilbaum t gilt, dass der Wurzelknoten von t die maximale Beschriftung in t hat. Geben Sie zuerst den Typ von `isHeap` an.

Übung 4 (AGS 12.1.27)

(a) Schreiben Sie in Haskell eine Funktion `incEntry :: [Int] -> Int -> [Int]`, so dass für jede Liste `l :: [Int]` und jede ganze Zahl `i :: Int` das Ergebnis der Funktionsanwendung `incEntry l i` die Liste ist, die aus `l` entsteht, indem der `i`-te Eintrag um 1 erhöht wird. Wenn die Liste `l` weniger als `i` Einträge hat, soll diese zuerst durch wiederholtes Anhängen von Einträgen mit dem Wert 0 auf die Länge `i` verlängert werden.
Beispiel: `incEntry [2, 3]  4 = [2, 3, 0, 1]`.
(b) Schreiben Sie in Haskell eine Funktion `rsum :: [Int] -> [Int]`, die zu einer Liste `l` eine Ausgabeliste gleicher Länge erzeugt, deren erster Eintrag der letzte Eintrag von `l`, deren zweiter (bzw. dritter) Eintrag die Summe der letzten beiden (bzw. drei) Einträge von `l`, u.s.w. ist.

Beispiel: `rsum [4, 2, 3, 8] = [8, 3+8, 2+3+8, 4+2+3+8] = [8, 11, 13, 17]`.
Sie dürfen hierzu die Funktion `reverse :: [Int] -> [Int]` zum Invertieren von Listen verwenden, die wie folgt definiert ist:

```
reverse [] = []
reverse (x:xs) = reverse xs ++ [x]
```

Hinweis: Sollten Sie in den Aufgabenteilen (a) oder (b) weitere Hilfsfunktionen nutzen, so sind diese ebenfalls vollständig zu definieren.

Zusatzaufgabe 1 (AGS 12.1.29)

(a) Geben Sie eine Funktion `test` an, die eine Liste ganzer Zahlen entgegen nimmt und auf folgende Eigenschaft prüft:
Die Differenz zwischen zwei benachbarten Elementen soll abwechselnd positiv und negativ sein, jedoch nicht null. Die Liste darf mit einer positiven oder negativen Differenz beginnen, sowie einelementig oder leer sein. Erfüllt die Liste diese Eigenschaft, so soll `test True` zurückgeben.

(b) Geben Sie eine Funktion `delcon :: Int -> [[Int]] -> [Int]` an, die aus einer Liste von Listen alle diejenigen Listen entfernt, deren Elemente in Summe die im ersten Argument gegebene ganze Zahl ergeben, und die verbleibenden Listen der Reihe nach zu einer Liste konkateniert.

Zusatzaufgabe 2 (AGS 12.1.28)

(a) Schreiben Sie in Haskell eine Funktion `dis :: [Int] -> Bool`, so dass für jede Liste `l :: [Int]` folgendes gilt: `dis l = True` genau dann, wenn kein Element in `l` mehrmals vorkommt.
Beispiel: `dis [5,2,6,4] = True` und `dis [4,6,2,6] = False`.

(b) Schreiben Sie in Haskell eine Funktion `g :: [Int] -> [Int]`, die von einer nichtleeren Liste `l` das kleinste in ihr vorkommende Element entfernt. Sie können annehmen, dass alle Elemente in `l` unterschiedlich sind.
Beispiel: `g [5, 2, 8, 3] = [5, 8, 3]`.