2. Aufgabenblatt zu Funktionale Programmierung vom 27.10.2009.
Fällig: 03.11.2009 / 10.11.2009 (jeweils 15:00 Uhr)

Themen: *Funktionen auf Zeichenreihen, ganzen Zahlen und Listen*

Für dieses Aufgabenblatt sollen Sie die zur Lösung der unten angegebenen Aufgabenstellungen zu entwickelnden Haskell-Rechenvorschriften in einer Datei namens `Aufgabe2.lhs` im home-Verzeichnis Ihres Gruppenaccounts auf der Maschine g0 ablegen. **Anders** als bei der Lösung zum ersten Aufgabenblatt sollen Sie also dieses Mal ein "**literate Script**" schreiben.

Versehen Sie wie schon in Ihrer Lösung zum ersten Aufgabenblatt auch hier alle Funktionen, die Sie zur Lösung benötigen, mit ihren Signaturen und kommentieren Sie Ihre Programme wieder aussagekräftig. Benutzen Sie, wo sinnvoll, Hilfsfunktionen und Konstanten. Im einzelnen sollen Sie folgende Problemstellungen bearbeiten:

1. Eine Liste ganzer Zahlen heißt *ausgezeichnet*, wenn kein Element mehr als einmal in der Liste enthalten ist. (Ausgezeichnete Listen können wir als Implementierungen von Mengen mithilfe von Listen auffassen (vgl. Teilaufgabe 4).)

   Schreiben Sie eine Haskell-Rechenvorschrift `auszeichnen` mit der Signatur `auszeichnen :: [Int] -> [Int]`, die angewendet auf eine Argumentliste l durch vollständiges Streichen aller mehr als einmal vorkommenden Elemente die eindeutig bestimmte ausgezeichnete Teilliste maximaler Länge r von l bestimmt. Dabei soll die relative Reihenfolge der Elemente von l in r erhalten bleiben.

2. Eine positive ganze Zahl heißt *perfekt*, wenn sie der Summe ihrer Teiler entspricht (mit Ausnahme der Zahl selbst). Beispielsweise ist die Zahl 6 perfekt, da für die Summe ihrer echten Teiler 1, 2 und 3 gilt: $1 + 2 + 3 = 6$.

   Schreiben Sie eine Haskell-Rechenvorschrift `perfekt` mit der Signatur `perfekt :: Integer -> Integer -> [Integer]`, die angewendet auf zwei Argumente m und n mit $0 < m \leq n$ alle perfekten Zahlen im Intervall von m bis n bestimmt (jeweils einschließlich). Die Resultatliste soll in diesem Fall aufsteigend geordnet sein. Erfüllen die Argumente die Bedingung $0 < m \leq n$ nicht, so soll als Resultat die einelementige Liste `[0]` ausgegeben werden.

3. Eine Relation $R \subseteq \mathbb{Z} \times \mathbb{Z}$ über den ganzen Zahlen $\mathbb{Z}$ heißt *antisymmetrisch*, wenn folgendes gilt:

   $$\forall\, x, y \in \mathbb{Z}.\; x\, R\, y \;\wedge\; y\, R\, x \Rightarrow x = y$$

   In Haskell lässt sich eine Relation über ganzen Zahlen wie folgt realisieren:

   ```
   type Relation = [(Int,Int)]
   ```

   Schreiben Sie eine Wahrheitswertfunktion `istAntisym` mit der Signatur `istAntisym :: Relation -> Bool`, die angewendet auf eine Relation `r` den Wahrheitswert `True` liefert, falls `r` antisymmetrisch ist, ansonsten den Wahrheitswert `False`.

4. Sei M eine endliche Menge. Die Menge aller Teilmengen von M heißt *Potenzmenge* von M.

   In Haskell können Mengen in Form ausgezeichneter Listen implementiert werden (vgl. Teilaufgabe 1).

   Schreiben Sie eine Haskell-Rechenvorschrift `pm` mit der Signatur `pm :: [Int] -> [[Int]]`, die angewendet auf eine endliche ausgezeichnete Argumentliste l die Potenzmenge von l, dargestellt als ausgezeichnete Liste von ausgezeichneten Listen, bestimmt. Die Reihenfolge der Listen in der Resultatliste ist dabei unerheblich. Ist die Argumentliste nicht ausgezeichnet, so liefert die Funktion das Resultat `[[],[]]`.

Denken Sie bitte daran, dass Sie für die Lösung dieses Aufgabenblatts ein "literate" Haskell-Skript schreiben sollen!