4. Aufgabenblatt zu Funktionale Programmierung vom 16.11.2005, fällig: 23.11.2005 (12:00 Uhr)

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

Für dieses Aufgabenblatt sollen Sie Haskell-Rechenvorschriften zur Lösung der unten angegebenen Aufgabenstellungen entwickeln und für die Abgabe in einer Datei namens `Aufgabe4.hs` ablegen. Sie sollen also wieder ein Standard-Haskell-Skript schreiben. Versehen Sie 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 folgende Problemstellungen bearbeiten:

1. Schreiben Sie eine Haskell-Rechenvorschrift `inject` mit der Funktionalität `inject :: Int -> [Int] -> [[Int]]`, die angewendet auf eine ganze Zahl und eine Liste ganzer Zahlen eine Liste von Listen ganzer Zahlen als Resultat liefert, die aus der Argumentliste so entstehen, dass das ganzzahlige Element an jeder Listenposition eingefügt wird. Folgende Beispiele zeigen die gewünschte Funktionalität: `inject 5 [1,2,3] == [[5,1,2,3],[1,5,2,3],[1,2,5,3],[1,2,3,5]]`, `inject 5 [] == [[5]]`

2. Zwei Zeichenreihen heißen *permutationsähnlich* genau dann, wenn jede der beiden Zeichenreihen eine Permutation der Zeichen der jeweils anderen Zeichenreihe ist. Zwei Zeichenreihen heißen *einzeichenähnlich* genau dann, wenn jede der Zeichenreihen durch Auslassen bzw. Einfügen genau eines Zeichens in die jeweils andere Zeichenreihe übergeht.

   - Schreiben Sie eine Wahrheitswertfunktion `permAehnlich` mit der Signatur `permAehnlich :: String -> String -> Bool`, die zwei Zeichenreihen auf Permutationsähnlichkeit überprüft. Ist diese gegeben, soll die Funktion den Wahrheitswert `True` liefern, sonst `False`.
   - Schreiben Sie eine Wahrheitswertfunktion `ezAehnlich` mit der Signatur `ezAehnlich :: String -> String -> Bool`, die zwei Zeichenreihen auf Einzeichenähnlichkeit überprüft. Ist diese gegeben, soll die Funktion den Wahrheitswert `True` liefern, sonst `False`.

3. In dieser Aufgabe schreiben wir eine Variante der Funktion `vorbereiteNaechste Auszaehlung` von Aufgabenblatt 3, die im Gesamtkontext der Auszählung zweckmäßiger zu verwenden wäre. Diese neue Funktion heiße `vna` und habe wie die entsprechende Funktion von Aufgabenblatt 3 die Signatur `vna :: [String] -> [[Int]] -> ([String],[[Int]])`. Wie ihr Vorbild von Aufgabenblatt 3 soll die Funktion `vna` zur Vorbereitung der nächsten Auszählung dienen, wenn in einem Wahlgang keiner der Kandidaten mit der absoluten Mehrheit der gültigen Stimmen gewählt worden ist. In diesem Fall werden der oder die Kandidaten mit den wenigsten Erststimmen vom Wahlvorschlag gestrichen, wobei deren relative Reihenfolge erhalten bleibt. Alle ungültigen Stimmzettel werden ausgeschieden und nicht ins Resultat übernommen, aus allen gültigen Stimmzetteln werden die Stimmen für die ausgeschiedenen Kandidaten entfernt

und die Stimmen für die verbliebenen Kandidaten an deren neuen Listenplatz im verkürzten Wahlvorschlag angepasst. Folgende Beispiele zeigen die Abbildungsweise der neuen Funktion `vna`:

```
vna ["John Smith","Judy Hall","John Doug"]
    [[2,1,3],[3,2,1],[3,1,2],[1,2,3],[2,3,1],[1,5]]
 == (["Judy Hall","John Doug"],[[1,2],[2,1],[2,1],[1,2],[1,2]])

vna ["John Smith","Judy Hall","John Doug"]
    [[1,2,3],[3,2,1],[3,1,2],[1,2,3],[2,3,1],[1,5]]
 == (["John Smith","John Doug"],[[1,2],[2,1],[2,1],[1,2],[2,1]])
```