3. Aufgabenblatt zu Funktionale Programmierung vom 04.11.2005, fällig: 16.11.2004

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

Für dieses Aufgabenblatt sollen Sie Haskell-Rechenvorschriften für die Lösung der unten angegebenen Aufgabenstellungen entwickeln und für die Abgabe in einer Datei namens `Aufgabe3.hs` ablegen. Wie für die Lösung zum ersten Aufgabenblatt sollen Sie dieses Mal also wieder 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. Im einzelnen sollen Sie die im folgenden beschriebenen Problemstellungen bearbeiten.

Wien hat unlängst gewählt. Auch in anderen Ländern wird in regelmäßigen Abständen gewählt, z.B. in Australien. Hier gibt es ein Mehrheitswahlrecht. Im ersten Wahlgang ist derjenige Kandidat gewählt, der die absolute Mehrheit der gültigen Stimmen auf sich vereinigen kann.

Von einem Wähler in Australien wird allerdings erwartet, sich bei der Stimmabgabe nicht nur für einen der angetretenen Kandidaten zu entscheiden, sondern alle zur Wahl angetretenen Kandidaten in einer Rangfolge zu reihen.

Die Auswertung der abgegebenen Stimmen geschieht dann wie folgt: Im ersten Schritt werden unter allen gültigen Stimmzetteln die Stimmen für die auf Rang 1 gesetzten Kandidaten ausgezählt. Hat dann einer der Kandidaten die absolute Mehrheit an Stimmen erreicht, so ist er gewählt und die Wahl abgeschlossen.

Konnte keiner der Kandidaten die absolute Mehrheit der gültigen Stimmen auf sich vereinigen, so werden der oder die Kandidaten mit den in der Auszählung wenigsten Stimmen bestimmt und ausgeschieden. Auf Stimmzetteln, deren Wähler einen dieser Kandidaten an Rang 1 gesetzt hatten, wird diese Position gestrichen, so dass jetzt der ursprünglich auf Rang 2 gesetzte Kandidat auf Rang 1 zu stehen kommt, der auf Rang 3 an Rang 2 usw. Jetzt beginnt die Auszählung unter den verbliebenen Kandidaten erneut, wobei wieder die nun an Rang 1 gesetzten Stimmen ausgezählt werden.

Dieser Vorgang wird solange wiederholt, bis entweder ein Kandidat mit der absoluten Mehrheit der gültigen Stimmen gewählt ist, oder alle Kandidaten ausgeschieden sind. In diesem Fall muss die gesamte Wahl wiederholt werden.

In dieser Aufgabe wollen wir Haskell-Rechenvorschriften schreiben, die eine Auswertung der Wahl nach obigem Muster erlauben.

1. Zunächst müssen wir gültige von ungültigen Stimmzetteln unterscheiden. Auf dem Wahlvorschlag sind die Namen der Kandidaten in einer bestimmten Reihenfolge aufgeführt. Die Zahl der Kandidaten bezeichnen wir mit n, wobei n mindestens 1 sein soll. Ein Stimmzettel ist dann und nur dann gültig, wenn auf ihm eine Folge von Zahlen von 1 bis n steht, in der jede dieser Zahlen genau einmal vorkommt. Bei 3 Kandidaten bedeutet die Folge 2, 1, 3, dass der Wähler den an Stelle 2 im Wahlvorschlag genannten Kandidaten auf Rang 1 setzt, den

an Stelle 2 genannten Kandidaten auf Rang 2 usw. Ein Stimmzettel der Form 2, 2, 3 wäre ungültig, ebenso ein Stimmzettel der Form 2, 1 oder der Form 3, 2, 1, 4. Für unsere Aufgabe modellieren wir Wahlvorschläge durch Listen von Zeichenreihen und Stimmzettel durch Listen von ganzen Zahlen. Schreiben Sie also eine Haskell-Rechenvorschrift `istGueltig` mit der Signatur `istGueltig :: [String] -> [Int] -> Bool`, die überprüft, ob ein abgegebener Stimmzettel im obigen Sinn gültig ist oder nicht. Der Aufruf von `istGueltig ["John Smith","Judy Hall","John Doug"] [2,3,1]` soll also den Wert `true` liefern, der Aufruf `istGueltig ["John Smith","Judy Hall","John Doug"] [1,2,5]` den Wert `false`. Ist der Wahlvorschlag leer, soll die Funktion `istGueltig` ebenfalls stets `false` liefern.

2. Schreiben Sie jetzt eine Haskell-Rechenvorschrift, die gültige von ungültigen Stimmzetteln trennt. Diese Rechenvorschrift heiße `trenneStimmen` und habe die Funktionalität `trenneStimmen :: [String] -> [[Int]] -> ([[Int]],[[Int]])`. In der ersten Komponente des Resultats sind dabei die gültigen Stimmabgaben, in der zweiten Komponente die ungültigen Stimmabgaben aufgeführt, und zwar in der gleichen relativen Reihenfolge, wie sie in der Liste aller abgegebenen Stimmen ("2. Argument") vorkommen. Der Aufruf von `trenneStimmen ["John Smith","Judy Hall","John Doug"] [[2,1,3],[3,4,1],[2,1],[1,2,3],[2,3,1]]` soll also das Resultat `([[2,1,3],[1,2,3],[2,3,1]],[[3,4,1],[2,1]])` liefern.

3. Schreiben Sie nun eine Haskell-Rechenvorschrift `auszaehlen`, die einen Teil des Auszählungsverfahrens simuliert. Die Funktion `auszaehlen` soll die Signatur `auszaehlen :: [String] -> [[Int]] -> [Int]` haben und für jeden im Wahlvorschlag genannten Kandidaten die Zahl der gültigen Stimmzettel liefern, in denen er auf Rang 1 gesetzt ist. Beachten Sie, dass die Liste der Stimmzettel sowohl gültige wie ungültige Stimmabgaben enthalten kann. Gezählt werden dürfen nur die gültigen. Der Aufruf von `auszaehlen ["John Smith","Judy Hall","John Doug"] [[2,1,3],[3,4,1],[2,1],[1,2,3],[2,3,1]]` soll also das Resultat `[1,2,0]` liefern.

4. Schreiben Sie jetzt eine Haskell-Rechenvorschrift `erfolgreich`, die feststellt, ob ein Kandidat im aktuellen Auszählungsgang mit der absoluten Mehrheit der gültigen Stimmen gewählt worden ist. Diese Funktion soll die Funktionalität `erfolgreich :: [String] -> [[Int]] -> Bool` haben und als Resultat `True` liefern, wenn ein Kandidat mit der absoluten Mehrheit der gültigen Stimmen gewählt ist, sonst `False`. Der Aufruf von `erfolgreich ["John Smith","Judy Hall","John Doug"] [[2,1,3],[3,4,1],[2,1],[1,2,3],[2,3,1]]` soll also das Resultat `True` liefern.

5. Schreiben Sie auch eine Haskell-Rechenvorschrift `gewaehltIst` mit der Signatur `gewaehltIst :: [String] -> [[Int]] -> String`. Ist ein Kandidat in der aktuellen Auszählung mit der absoluten Mehrheit der gültigen Stimmen gewählt, so ist das Resultat der Name des gewählten Kandidaten, ansonsten die leere Zeichenreihe. Der Aufruf von `gewaehltIst ["John Smith","Judy

`Hall","John Doug"] [[2,1,3],[3,4,1],[2,1],[1,2,3],[2,3,1]]` soll also das Resultat `"Judy Hall"` liefern.

6. Ist eine Auszählung nicht erfolgreich, d.h. kein Kandidat mit der absoluten Mehrheit der gültigen Stimmen gewählt, muss für den nächsten Auszählungsgang der Wahlvorschlag um die Namen der mit den wenigsten Erststimmen ausgeschiedenen Kandidaten verkürzt werden und die Stimmzettel, die einen dieser Kandidaten an erster Stelle gereiht hatten, durch Nachrücken wie eingangs beschrieben vor der nächsten Auszählung angepasst werden. Schreiben Sie eine entsprechende Haskell-Rechenvorschrift `vorbereiteNaechsteAuszaehlung` mit der Signatur `vorbereiteNaechsteAuszaehlung :: [String] -> [[Int]] -> ([String],[[Int]])`, die das leistet. Die relative Reihenfolge der Kandidaten soll dabei in der verkürzten Liste beibehalten werden, die Liste der Stimmzettel soll nur noch gültige Stimmzettel enthalten, ebenfalls in derselben relativen Reihenfolge wie im Argument. Ist ein gültiger Stimmzettel anzupassen, rückt der an erster Stelle der Liste genannte Kandidat an die letzte Stelle der Liste, damit der Stimmzettel insgesamt seine formale Gültigkeit behält. Ist Kandidat 3 also ausgeschieden, wird ein Stimmzettel, der Kandidat 3 an erster Stelle gereiht hatte, wie folgt geändert: `[3,2,1,4]` geht über in `[2,1,4,3]`.

7. Schreiben Sie jetzt eine Haskell-Rechenvorschrift `endergebnis`, die die Gesamtauswertung einer Wahl nach eingangs beschriebenem Muster liefert. Die Funktion `endergebnis` hat die Funktionalität `endergebnis :: [String] -> [[Int]] -> String`. Enthält der Wahlvorschlag weniger als zwei Namen, so liefert die Funktion `endergebnis` das Resultat `"Ungueltiger Wahlvorschlag"`. Ansonsten liefert die Funktion `endergebnis` als Resultat den Namen des Kandidaten, der erstmals in einem der Auszählungsgänge mit der absoluten Mehrheit der gültigen Stimmen gewählt ist, wenn es einen solchen gibt, ansonsten die Zeichenreihe `"Kein Kandidat erfolgreich"`. Wir dürfen uns darauf verlassen, dass keiner der Kandidaten eines Wahlvorschlags auf einen der Namen `"Ungueltiger Wahlvorschlag"` oder `"Kein Kandidat erfolgreich"` hört. Der Aufruf von `endergebnis ["John Smith","Judy Hall","John Doug"] [[2,1,3],[3,4,1],[2,1],[1,2,3],[2,3,1]]` soll also das Resultat `"Judy Hall"` liefern, der Aufruf `endergebnis ["John Smith","Judy Hall","John Doug"] [[2,1,3],[3,4,1],[2,1],[1,2,3],[3,2,1]]` das Resultat `"Kein Kandidat erfolgreich"`.

8. Zum Abschluss modifizieren wir das Wahlrecht um ein Quorum. Ein Kandidat gilt nur dann als gewählt, wenn er neben der absoluten Mehrheit an gültigen Stimmen in einem der Auszählungsgänge dabei mindestens einen bestimmten Prozentsatz der insgesamt abgegebenen gültigen oder ungültigen Stimmen erreicht hat. Wir schreiben dazu eine Funktion `quEndergebnis` mit der Signatur `quEndergebnis :: Int -> [String] -> [[Int]] -> (String,Int,Int)`. Dabei gibt das erste Argument das zu erreichende Quorum an (einen Prozentsatz zwischen 0 und 100.) Im Resultattripel enthält die erste Komponente den Namen des gewählten Kandidaten, so einer gewählt ist, in der zweiten Komponente das zu erreichende Quorum (also den Wert des ersten Arguments) und in der

dritten Komponente den tatsächlich erreichten Prozentanteil an abgegebenen Stimmen schulmäßig gerundet auf volle Prozent. Enthält ein Wahlvorschlag weniger als einen Namen, liefert die Funktion das Resultat `("Ungueltiger Wahlvorschlag",0,0)`, ist kein Kandidat erfolgreich, das Resultat `("Quorum verfehlt",m,n)`, wobei `m` das zu erreichende Quorum ist und `n` der schulmäßig auf volle Prozent gerundete Stimmenanteil des oder der relativ erfolgreichsten Kandidaten. Liegt das zu erreichende Quorum nicht zwischen 0 und 100, liefert die Funktion das Resultat `("Ungueltiges Quorum",m,0)`, wobei `m` das vorgegebene Quorum ist. Der Aufruf von `quEndergebnis 50 ["John Smith","Judy Hall","John Doug"] [[2,1,3],[3,4,1],[2,1],[1,2,3],[2,3,1]]` soll also das Resultat `("Quorum verfehlt",50,40)` liefern, der Aufruf `quEndergebnis 33 ["John Smith","Judy Hall","John Doug"] [[2,1,3],[3,4,1],[2,1],[1,2,3],[2,3,1]]` das Resultat `("Judy Hall",33,40)`.