Praktische Informatik 3 WS 06/07

3. Übungsblatt

Ausgabe: 28.11.06

Bearbeitungszeit: Zwei Wochen

Christoph Lüth <cxl>
Matthias Berger <tokio>
Klaus Hartke <hartke>
Cui Jian <ken>
Friederike Jolk <rikej>
Christian Maeder <maeder>
Diedrich Wolter <dwolter>

7 *What Type is it?* *3 Punkte*

Geben Sie Typableitungen wie in der Vorlesung für folgende Funktionen an:

1. `f x = filter ("3" ==). (map (x ++))`
2. `g x y = (x < y) && (any (3 ==) x)`
3. `funny = (.).(.)`

8 *Der Haskell Kalender* *7 Punkte*

> Wenn der eifrige Student
> mittwochs in den Hörsaal rennt,
> hat er die Vorlesung verpennt.

Damit das nicht passiert (und man immer weiß, wann Dienstag ist), entwickeln wir in dieser Aufgabe ein kleines Werkzeug, welches für einen gegebenen Monat die Aufteilung der Tage auf Wochen anzeigt, analog dem Unix-Hilfsprogramm `cal`. Dieses erzeugt zum Beispiel für `cal 11 2006` folgende Ausgabe:

```
   November 2006
Su Mo Tu We Th Fr Sa
          1  2  3  4
 5  6  7  8  9 10 11
12 13 14 15 16 17 18
19 20 21 22 23 24 25
26 27 28 29 30
```

Entwickeln Sie eine Funktion, die einen String zurückgibt, der diese Ausgabe realisiert:

```
type Year  = Int
type Month = Int
cal :: Year-> Month-> String
```

Hierbei ist `cal` für alle Jahre ab einschließlich dem Jahr 1 definiert.

Die Funktion kann in erster Annäherung durchgängig (für alle Monate und Jahre) den *gregorianischen Kalender* annehmen (siehe unten). Hierbei erfolgt die Berechnung der Wochentage nach folgenden Regeln:

- Der 1. Januar des Jahres 1 war ein Montag.
- Jedes Jahr hat 365 Tage, außer Schaltjahre, welche 366 Tage haben.
- Schaltjahre sind alle durch 4 ganzzahlig teilbaren Jahre, bis auf durch 100 ganzzahlig teilbare Jahre, aber eingeschlossen durch 400 ganzzahlig teilbare Jahre.

Der gregorianische Kalender ist eine Verfeinerung des ursprünglichen *julianischen* Kalenders, für den die Berechnung der Wochentage nach ähnlichen Regeln erfolgt, bis auf folgende Änderungen:

- Der 1. Januar des Jahres 1 war ein Sonnabend.
- Schaltjahre sind alle durch 4 ganzzahlig teilbaren Jahre.

Der julianische Kalender wurde von dem römischen Informatik-Pionier Julias Cäsar (Gründer der Firma Olivetti) eingeführt. Aufgrund seiner mangelnden Genauigkeit wurde er nach dem 18. Februar 1700 vom gregorianischen Kalender abgelöst[1]; auf den 18. Februar 1700 folgt also unmittelbar der 1. März 1700.

Erweitern Sie Ihre Funktion so, dass sie diese Datumsumstellung korrekt berücksichtigt, d.h. unter anderem folgende Ausgaben erzeugt:

```
    February 100
Su Mo Tu We Th Fr Sa
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
```

```
   February 1700
Su Mo Tu We Th Fr Sa
                1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18
```

Punkteverteilung für die Aufgabe:

1. Datumsberechung nach dem gregorianischen Kalender (*6 Punkte*)
2. Historisch korrekte Berücksichtigung des julianischen Kalenders (*1 Punkte*)

[1] Jedenfalls in Bremen. Aus politischen Gründen setzte sich der Kalender in Europa erst allmählich durch; in England und seine Kolonien wurde der gregorianische Kalender beispielsweise erst am 2. September 1752 eingeführt.

9 *Haskell Word* *10 Punkte*

Langsam nähern wir uns dem Kern der Informatik, nämlich Textverarbeitung und Tabellenkalkulationsprogrammen. Wir fangen mit dem ersten an, denn Texte sind Zeichenketten, und damit können wir schon gut umgehen. Die Eroberung des Weltmarktes ist somit nur noch eine Frage der Zeit.

Unser Textverarbeitungsprogramm `HaskellWord` modelliert ein Dokument als eine Folge von Absätzen (wobei ein Absatz ein String ist). Eine *Einfügemarke* teilt das Dokument in zwei Teile, den Teil vor der Marke (welcher *von hinten nach vorne* repräsentiert wird, um schnelles Einfügen und Auslesen zu ermöglichen), und ein Teil danach:

```
type Doc = (Text, Text)
type Text = [Paragraph]
type Paragraph = String
```

Wir implementieren zuerst Funktionen, dies es erlauben, den Text zu editieren:

```
forward  :: Doc-> Doc
backward :: Doc -> Doc
move     :: Int-> Doc -> Doc

insert   :: String-> Doc-> Doc
insBreak :: Doc-> Doc

del   :: Doc-> Doc
bs    :: Doc-> Doc
erase :: Int-> Doc-> Doc
```

- `forward`, `backward` und `move` bewegen die Einfügemarke.
- `insert` fügt einen Text ein und bewegt die Eingabemarke *hinter* diesen Text. Aus dem eingegebenen Text sollen alle Zeilenvorschübe, Rückläufe und nicht druckbaren Zeichen entfernt werden.
- `insBreak` fügt einen Absatz ein.
- `del` und `bs` löschen das Zeichen vor bzw. hinter der Eingabemarke; `erase` löscht mehrere Zeichen.

Zusätzlich zum Editieren soll der Text auch formatiert ausgegeben werden. Dazu implementieren wir die Funktion

```
format :: (Int, Bool) -> Doc-> String
```

welche den Text in Zeilen der angegebenen Länge formatiert ausgibt. Der Wahrheitswert gibt an, ob die Zeilen im Blocksatz (`True`) oder Flattersatz (`False`) ausgegeben werden sollen.

Hierzu bietet sich folgende Vorgehensweise an:

1. Zuerst wird aus dem Dokument eine einheitliche Liste von Absätzen gemacht;
2. Dann wird jeder Absatz einzeln formatiert;
3. Zum Schluss werden alle formatierten Texte zusammengehängt (mit einem Zeilenvorschub dazwischen).

Um einen einzelnen Absatz zu formatieren,

1. brechen wir erst den Absatz in einzelne Worte auf,
2. unterteilen die Liste der Worte in Listen von Listen von Worten, die jeweils zusammen mit den dazwischen stehenden Leerzeichen nicht länger als die gegebene Zeilenlänge sind,
3. fügen dann in jeder dieser Listen die Worte wieder zu einer Zeile zusammen, wobei gegebenenfalls mehr Leerzeichen für den Blocksatz eingefügt werden müssen (außer für die letzte Zeile eines Absatzes, die auch im Blocksatz linksbündig gesetzt wird),
4. und fügen zum Schluss alle Zeilen zu einer Zeichenkette zusammen.

Jetzt fehlt nur noch die Markteinführung, die rechtzeitig zum Weihnachtsgeschäft abgeschlossen sein sollte (s. Abgabedatum).

Dies ist Version 1.0 vom 2006/11/28 13:46:29.