Universität Bremen

Praktische Informatik 3 WS 18/19

Christoph Lüth
Thomas Barkowsky
Andreas Kästner
Gerrit Marquardt
Tobias Haslop
Matz Habermann
Berthold Hoffmann

7. Übungsblatt

Ausgabe: 27.11.18
Abgabe: 04.12.18 12:00

Dieses ist ein komplettes *Bonusübungsblatt*. Die in diesem Blatt erbrachten Punkte werden zu den bisherigen dazu gerechnet. Sie haben damit die Möglichkeit, verpaßte oder misslungene Abgaben auszugleichen, nochmal etwas Haskell zu üben, oder sich vieleicht auch nur mit Niveau etwas Zeit zu vertreiben.

7.1 *Links, Rechts, Mitte* *5 Punkte*

In dieser Übung wollen wir Text formatieren, und zwar linksbündig, rechtsbündig oder auf beiden Seiten bündig. Der Einfachheit formatieren wir den ganzen Text in einem Absatz.

Dabei gehen wir wie folgt vor:

1. Wir definieren zuerst einen Datentyp für Formatierung:

   ```
   data Format = Flushleft | Justify | Flushright
   ```

2. Um einen Text zu formatieren, müssen wir ihn zuerst in einzelne Wörter zerlegen (vordefinierte Funktion `words`). Diese Liste von Wörtern wird jetzt zu Gruppen von Wörtern zusammengefasst, so dass jede einzelne Gruppe nicht länger als eine Zeile ist (d.h. die Summe der Länge der Wörter plus ihre Anzahl minus eins ist kürzer oder gleich der gewünschten Zeilenlänge):

   ```
   chunks :: Int→ [String]→ [[String]]
   ```

   Dazu ist es sinnvoll, eine lokale Hilfsfunktion `oneChunk` zu definieren, welche aus einer Liste von Wörtern rekursiv so so lange das erste abnimmt, wie die Zeilenlänge noch nicht überschritten ist.[1] `oneChunk` gibt die abgenommen Wörter sowie die restlichen Wörter zurück. Jetzt muss nur noch `oneChunk` wiederum rekursiv aufgerufen werden, bis die restlichen Wörter leer sind.

   Wörter, die länger sind als die spezifizierte Zeilenlänge, erzeugen Zeilen mit Überlänge:

   ```
   chunks 5 ["ab", "cd", "edfghij", "klm"] ⇝
   [["ab","cd"],["edfghij"],["klm"]]
   ```

   Dieses ist die einzige Situation, in der eine Zeile länger als die spezifizierte Länge ist.

3. Für die Formatierung einer einzelnen Zeile implementieren Sie

   ```
   formatLine :: Format→ Int→ [String]→ String
   ```

   welche eine Liste von Wörtern wie angegeben formatiert zurückgibt.

   Für links- bzw. rechtsbündige Formatierung werden die Wörter mit Leerzeichen verbunden zusammengefügt (`unwords`), und mit Leerzeichen links bzw. rechts auf die gewünschte Länge aufgefüllt.

   Lediglich die bündige Formatierung (`Justify`) ist etwas delikat. Hierzu müssen Sie geeignet Leerzeichen zwischen die Wörte einfügen, und zwar so dass:

   (i) der Unterschied in der Anzahl der Leerzeichen höchsten eins beträgt, und

   (ii) weiter links in der Zeile nie mehr Leerzeichen zwischen den Wörtern sind als rechts.

[1] Dieses ist ein sogenannter "greedy"-Algorithmus, und es sei nicht verschwiegen, dass er in manchen Situationen zu ästethisch unschönen Zeilenumbrüchen führt, aber das soll uns hier nicht weiter beschäftigen.

Betrachten wir ein Beispiel. Wir wollen folgende Wörter auf 40 Zeichen bündig formatieren:

```
l = ["Lorem", "ipsum", "dolor", "sit", "amet,", "consetetur"]
```

Diese Wörter haben insgesamt eine Länge von 33 Zeichen, wir müssen als 7 Leerzeichen in 5 Lücken (Länge von l minus 1) einfügen. Das sind mindestes 1 Leerzeichen pro Lücke ($7 \div 5$). Es bleiben noch 2 ($7 \bmod 5$) Leerzeichen, die wir in die letzten beiden Lücken einfügen. Sei g die Anzahl der Lücken, und s die Anzahl der gesamt zu verteilenden Leerzeichen, dann müssen also in die i-te Lücke (gezählt ab 0) $spc(i)$ Leerzeichen eingefügt werden, wobei spc wie folgt definiert ist:

$$spc(i) = \begin{cases} s \div g + 1 & i \geq g - s \bmod g \\ s \div g & \text{sonst} \end{cases}$$

Für unser Beispiel ergibt sich

```
formatLine Justify 40 l ⇝ "Lorem␣ipsum␣dolor␣sit␣␣amet,␣␣consetetur"
```

4. Mit chunks und formatLine implementieren wir die Hauptfunktion

   ```
   format :: Format→ Int→ String→ String
   ```

   welche einen Text wie angegeben formatiert. Dazu wird der Text mit chunks gruppiert, jede einzelne Gruppe (entspricht einer Zeile) mit formatLine formatiert, und das ganze mit unlines wieder zusammengesetzt.

   Für einen längeren Beispieltext ergibt sich:

   ```
   l = "Lorem␣ipsum␣dolor␣sit␣amet,␣consetetur␣sadipscing␣elitr,␣sed␣diam
   nonumy␣eirmod␣tempor␣invidunt␣ut␣labore␣et␣dolore␣magna␣aliquyam␣erat,␣sed
   diam␣voluptua.␣At␣vero␣eos␣et␣accusam␣et␣justo␣duo␣dolores␣et␣ea␣rebum."

   putStrLn $ format 40 Flushleft l ⇝
   Lorem ipsum dolor sit amet, consetetur
   sadipscing elitr, sed diam nonumy
   eirmod tempor invidunt ut labore et
   dolore magna aliquyam erat, sed diam
   voluptua. At vero eos et accusam et
   justo duo dolores et ea rebum.

   putStrLn $ format 40 Justify l ⇝
   Lorem ipsum dolor sit amet, consetetur
   sadipscing elitr, sed diam nonumy
   eirmod tempor invidunt ut labore et
   dolore magna aliquyam erat, sed diam
   voluptua. At vero eos et accusam et
   justo duo dolores et ea rebum.

   putStrLn $ format 40 Flushright l ⇝
     Lorem ipsum dolor sit amet, consetetur
          sadipscing elitr, sed diam nonumy
        eirmod tempor invidunt ut labore et
      dolore magna aliquyam erat, sed diam
        voluptua. At vero eos et accusam et
             justo duo dolores et ea rebum.
   ```

   Hinweise: Normalerweise wird die letzte Zeile gesondert behandelt und nicht bündig gesetzt, das brauchen Sie hier nicht zu machen.

   Falls Ihnen die bündige Formatierung Schwierigkeiten bereitet, implementieren Sie zuerst links- und rechtsbündige Formatierung.