Universität Bremen

Praktische Informatik 3 WS 2023/24

Thomas Barkowsky
Tede von Knorre
Yaroslav Purgin
Tim Sperling

4. Übungsblatt

Ausgabe: 13.11.2023
Abgabe: 20.11.2023

In diesem Übungsblatt schreiben wir einige Funktionen zur Textanalyse und -formatierung.

4.1 *Textstatistik* *4 Punkte*

Wir beginnen mit einer ganz einfachen Funktion, die für uns die Anzahl Zeichen in einem gegebenen Text ermittelt. Ein Text ist hier einfach als String repräsentiert:

```
countChars "The quick brown fox jumps over the lazy dog." ⇝ 44
```

Nun sind die Leerzeichen in einem Text möglicherweise nicht wirklich interessant. Daher benötigen wir auch noch die Funktion

```
countCharsWOSpaces :: String → Int ,
```

die alle Zeichen mit Ausnahme der Leerzeichen zählt:

```
countCharsWOSpaces "The quick brown fox jumps over the lazy dog." ⇝ 36
```

Ebenfalls interessant ist die Wortanzahl in einem gegebenen Text:

```
countWords "The quick brown fox jumps over the lazy dog." ⇝ 9
```

Die Funktion `words` (*standard prelude*) könnte bei der Realisierung von `countWords` hilfreich sein.

Was ist noch von Interesse? Wir wäre es mit der durchschnittlichen Länge eines Wortes in einem gegebenen Text:

```
averageWordLength "The quick brown fox jumps over the lazy dog." ⇝ 4.0
```

Für die Konvertierung von `Int` Werten in Fließkommazahlen werdet ihr hier vermutlich die Funktion `fromIntegral` benötigen.

Habt ihr euch mal gefragt, warum ins Deutsche übersetzte längere Texte immer deutlich länger sind als ihr englisches Pendant? Das liegt u.a. an der unterschiedlichen durchschnittlichen Wortlänge in den beiden Sprachen. In der Vorlage findet ihr zwei Beispieltexte, einen in deutscher Übersetzung, den anderen im englischen Original:

```
averageWordLength text_en ⇝ 3.9672668
averageWordLength text_dt ⇝ 4.91433
```

Schließlich wollen wir noch zählen, wie oft die Buchstaben des Alphabets in einem gegebenen Text vorkommen.

```
freqOfChars "The quick brown fox jumps over the lazy dog." ⇝
[('O',4),('E',3),('H',2),('R',2),('T',2),('U',2),('A',1),('B',1),('C',1),('D',1),('F',1),('G',1),('I',1),
('J',1),('K',1),('L',1),('M',1),('N',1),('P',1),('Q',1),('S',1),('V',1),('W',1),('X',1),('Y',1),('Z',1)]
```

Die zurückgegebene Liste soll hierbei folgende Kriterien erfüllen:

- Es sollen genau die 26 Buchstaben des Alphabets von A bis Z enthalten sein.
- Alle Buchstaben werden als Großbuchstaben aufgeführt. Die Funktionen `toUpper` ist hierfür sicher nützlich.
- Die Liste soll nach absteigender Häufigkeit sortiert sein, d.h. der am häufigsten auftretende Buchstabe steht am Anfang der Liste. In `Data.List` (in der Vorlage importiert) findet ihr nützliche Funktionen zum Sortieren.
- Wenn Buchstaben in gleicher Häufigkeit auftreten, dann sollen diese in der Liste alphabetisch sortiert sein (siehe bspw. die Buchstaben H, R, T und U in obigem Beispiel).

Hier noch ein Beispiel für einen längeren Text:

```
freqOfChars text_dt ⇝
[('E',493),('N',284),('I',263),('S',216),('A',212),('T',180),('R',178),('H',167),
('D',155),('U',124),('C',122),('L',116),('M',109),('G',86),('W',85),('O',79),
('B',55),('F',30),('K',27),('V',24),('Z',19),('P',14),('Y',7),('J',4),('Q',1),('X',0)]
```

4.2 *Texte mit Format* *6 Punkte*

Im zweiten Teil dieses Übungsblatts bringen wir unsere Texte ein wenig in Form. Wir beginnen mit einer Funktion, die einen gegebenen Text (repräsentiert als String) in eine Liste von Teilstrings (oder Zeilen) mit einer vorgegebenen maximalen Anzahl von Zeichen zerlegt:

```
formatText "The␣quick␣brown␣fox␣jumps␣over␣the␣lazy␣dog." 15 ⇝
["The␣quick␣brown","fox␣jumps␣over","the␣lazy␣dog."]
```

Wenn die vorgegebene Zeilenlänge kürzer ist als das längste Wort in dem Text, so soll dieser Parameter so angepasst werden, dass auch das längste Wort genau in eine Zeile passt:

```
formatText "The␣quick␣BigBrownFox␣jumps␣over␣the␣lazy␣dog." 5 ⇝
["The␣quick","BigBrownFox","jumps␣over","the␣lazy","dog."]
```

Wenn wir uns einem in Zeilen formatierten Text zurechtfinden wollen, dann sind Zeilennummern hilfreich. Die Funktion

```
addLineNrs :: [String] → [String]
```

fügt am Anfang einer jeden Zeile eines Textes (repräsentiert als Liste von Strings) eine dreistellige Zeilennummer, gefolgt von einem Leerzeichen ein. Die Zeilennummern beginnen bei 1, Zeilennummern mit weniger als drei Stellen werden mit führenden Nullen aufgefüllt:

```
addLineNrs ["The␣quick␣brown","fox␣jumps␣over","the␣lazy␣dog."] ⇝
["001␣The␣quick␣brown","002␣fox␣jumps␣over","003␣the␣lazy␣dog."]
```

Jetzt wollen wir noch wissen, in welcher Zeile unseres Textes ein bestimmter Teilstring zu finden ist. Die Funktion

```
searchString :: String → [String] → [String]
```

liefert alle Zeilen zurück, die den gesuchten String enthalten:

```
searchString "fox" ["001␣The␣quick␣brown","002␣fox␣jumps␣over","003␣the␣lazy␣dog."] ⇝
["002␣fox␣jumps␣over"]
```

Zu guter Letzt wollen wir unsere so erzeugten Texte wohlformatiert mit putStrLn auf der Konsole ausgeben können. Die Funktion

```
nicelyPrint :: [String] → String
```

erzeugt aus einer Liste von Strings einen einzelnen String, bei dem jede Zeile (also jeder String der übergebenen Liste) mit einem *newline* ("\n") abgeschlossen wird:

```
putStrLn $ nicelyPrint ["001␣The␣quick␣brown","002␣fox␣jumps␣over","003␣the␣lazy␣dog."] ⇝
001 The quick brown
002 fox jumps over
003 the lazy dog.
```

Mit diesen Funktionen gerüstet können wir nun z.B. unseren englischen Text bearbeiten:

```
putStrLn $ nicelyPrint $ searchString "Moses" $ addLineNrs $ formatText text_en 80 ⇝
029 supper she got out her book and learned me about Moses and the Bulrushers, and I
031 Moses had been dead a considerable long time; so then I didn't care no more
036 about it. Here she was a-bothering about Moses, which was no kin to her, and no
```

(Bei einer Formatierung auf 80 Zeichen pro Zeile wird in den Zeilen 29, 31 und 36 "Moses" erwähnt.)