PHILIPPS-UNIVERSITÄT MARBURG
Fachbereich Mathematik und Informatik
Prof. Dr. R. Loogen

D-35032 Marburg
Hans Meerwein Straße
Lahnberge

17. Dezember 2003

Übungen zur „Praktischen Informatik III“, WS 2003/04

Nr. 9, Besprechung bzw. Abgabe: 14. und 15. Januar in den Übungen

---

A. Mündliche Aufgaben

41. Segmentierung von Listen

    (a) Schreiben Sie eine Funktion `segments :: (a -> Bool) -> [a] -> [[a]]`, die eine Liste in Teillisten zerlegt, deren Elemente ein gegebenes Prädikat erfüllen.
    Beispiel: `segments isDigit "100:200:x300" =>* ["100","200","","300"]`

    (b) Definieren Sie die Prelude-Funktionen `lines` und `words` mit `segments`.

42. Geben Sie Funktionen mit folgenden Typen an:

    (a) `a -> Bool` (b) `Bool -> a` (c) `a -> a -> b -> (b,a)` (d) `a -> b`

B. Hausaufgaben

43. Wortzähler — 3 Punkte

    (a) Definieren Sie eine Funktion `groupBy :: (a -> a -> Bool) -> [a] -> [[a]]`, die beim Aufruf `groupBy p xs` die Liste `xs` so in Segmente `[x1,x2,...]` einteilt, dass für jedes `i (p x1 xi)` wahr ist. — / 1

    (b) Schreiben Sie eine Funktion `wordcount :: String -> [(String, Int)]`, die zu einem Text eine Liste der darin vorkommenden Wörter zusammen mit der Anzahl der Vorkommen ausgibt. — / 2
    Beispiel: `wordcount "Jingle Bells Jingle" =>* [("Bells",1),("Jingle",2)]`

44. Dekodierung — 6 Punkte

    Die folgende Funktion `caesar` (siehe Datei `bspHOF.hs` auf der Vorlesungsseite) verschlüsselt Texte nach der Caesar-Methode, d.h. durch eine zyklische Alphabetverschiebung um eine vorgegebene ganze Zahl.

```
caesar       :: Int -> String -> String
caesar n str =  map (code n) str

code                :: Int -> Char -> Char
code n x | letter x  =  xox
         | otherwise =  x
    where  ox  =  ord x + n
           xox |    (upperLetter x && ox > ord 'Z')
               || (lowerLetter x && ox > ord 'z') = chr (ox - 26)
               | otherwise                        = chr ox
```

(a) Schreiben Sie eine Funktion `decaesar :: Int -> String -> String`, die eine Dekodierung einer Caesarkodierung mit vorgegebener Verschiebungszahl durchführt. / 1

(b) Implementieren Sie eine Funktion / 2

`codeBreakerDic :: String -> [String] -> String`,

die eine mit der Caesarmethode verschlüsselte Zeichenkette dekodiert. Das zweite Argument dient als Verzeichnis von Wörtern im Klartext, von denen angenommen wird, dass sie im dekodierten Text auftreten. Sie können davon ausgehen, dass ein Text korrekt dekodiert ist, wenn ein Wort im vorgegebenen Verzeichnis auftritt. Ist eine Dekodierung nicht möglich, soll die leere Zeichenkette ausgegeben werden.

(c) Definieren Sie eine Funktion `codeBreakerFreq :: String -> String`, die eine Zeichenkette in Caesarverschlüsselung dekodiert, wobei die Verschiebung dadurch ermittelt wird, dass der häufigste Buchstabe im kodierten Text mit 'e' bzw. 'E' identifiziert wird. / 2

Verwenden Sie die in Aufgabe 36 (Übungsblatt 8) entwickelte Funktion `frequency`, um den häufigsten Buchstaben eines Textes zu bestimmen.

(d) Testen Sie Ihre Funktionen mit dem folgenden Text (siehe Datei `nachricht.txt` auf der Vorlesungsseite): / 1

```
Gzdss dhmdq ztbg ezrs ldgq Udqrszmc
zkr vhd chd cqdh Vdhrdm ztr Lnqfdmkzmc
tmc khdrrd rhbg ctdmjdm, dq vzdq vngk mhd
cdl Rsdqmkdhm mzbgfdqdhrs vhd rhd;
cdmmnbg, vdmm mtm czr Vdhgmzbgsredrs
rdhmd Khbgskdhm vnmmhfkhbg rbgdhmdm kzdrrs,
ezdkks ztbg zte rdhm udqrszdmchf Fdrhbgs,
dq lzf dr ldqjdm ncdq mhbgs,
dhm eqdtmckhbgdq Rsqzgk
cdr Vtmcdqrsdqmdr unm czytlzk.
(Vhkgdkl Atrbg)
```

45. Weihnachtsaufgabe 3 Punkte

Schreiben Sie eine Funktion `christmasTree :: String -> IO()`, die einen Text in Tannenform ausgibt:

Allen
Hörerinnen
und Hörern ein
gesegnetes Weihnachtsfest
und ein gutes Jahr
2004!