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

D-35032 Marburg
Hans Meerwein Straße
Lahnberge

10. Dezember 2003

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

Nr. 8, Besprechung bzw. Abgabe: 17. und 18. Dezember in den Übungen

---

A. Mündliche Aufgaben

36. Buchstabenhäufigkeit

(a) Schreiben Sie eine Funktion zum Mischsortieren (siehe Aufgabe 12), bei der die Mischfunktion als Argument übergeben wird:

```
mergeSort :: ([t] -> [t] -> [t]) -> [t] -> [t]
```

(b) Definieren Sie unter Verwendung von `mergeSort` eine Funktion

```
sortfold :: [(Char, Int)] -> [(Char, Int)]
```

die eine Liste von Paaren aus je einem Buchstaben und einer Zahl nach den Buchstaben sortiert und dabei Paare mit gleichen Buchstaben durch die Addition der Zahlen zusammenfasst.

Beispiel:
```
sortfold [('c',2), ('d',1), ('a',4), ('d',2), ('c',5)]
    ==> [('a',4), ('c',7), ('d',3)]
```

(c) Implementieren Sie unter Verwendung der in (a) und (b) definierten Funktionen die Funktion

```
frequency :: [Char] -> [(Char, Int)]
```

die zu einer gegebenen Zeichenkette eine Liste aller darin vorkommenden Zeichen zusammen mit der Häufigkeit ihres Auftretens erstellt. Die Ergebnisliste soll dabei nach der Häufigkeit sortiert sein.

Beispiel:
```
frequency "battat"
    ==> [('b',1), ('a',2), ('t',3)]
```

37. Rosenbäume

Ein Rosenbaum ist ein Baum, dessen Knoten nicht notwendigerweise dieselbe Anzahl an Teilbäumen besitzen. Blätter werden durch Knoten repräsentiert, die keine Teilbäume haben:

```
data Rose a = Node a [Rose a]
```

Implementieren Sie die folgenden Funktionen:

(a) `degree :: Rose a -> Int`
ermittelt den maximalen Verzweigungsgrad eines Rosenbaums. Dabei handelt es sich um die größte Anzahl von Teilbäumen, die ein Knoten des Rosenbaums besitzt. Im Beispiel zu Teil (b) ist der maximale Verzweigungsgrad 3.

(b) `toB :: Rose a -> BTree a`
transformiert einen Rosenbaum in einen binären Baum vom Typ `BTree a`:

```
data BTree a = Leaf a  | Fork (BTree a) (BTree a)
```

Dabei soll gemäß dem folgenden Beispiel vorgegangen werden:

```
    1                        o
   /|\                      / \
  2 3 4      --->          o   4
   / \                   /   \
  5   6                 o     o
                       / \   / \
                      1   2 o   6
                             / \
                            3   5
```

(c) `dfo :: Rose a -> [a]` traversiert einen Rosenbaum in Tiefensuche.

(d) `bfo :: Rose a -> [a]` traversiert einen Rosenbaum in Breitensuche.

B. Hausaufgaben

38. Binärbäume **4 Punkte**

Gegeben seien die folgenden Definitionen:

```
data BBaum a = Empty | Node a (BBaum a) (BBaum a)
foldTree :: (a -> b -> b -> b) -> b -> BBaum a -> b
foldTree f e Empty        = e
foldTree f e (Node x l r) = f x (foldTree f e l) (foldTree f e r)
```

(a) Definieren Sie unter Verwendung der Funktion `foldTree` eine Funktion `size :: BBaum a -> Int`, die die Anzahl der Knoten eines Binärbaums bestimmt. / 1

(b) Schreiben Sie eine Haskell-Funktion `allTrees :: Int -> [BBaum ()]`, die zu einer Zahl n eine Liste aller Binärbäume `t :: BBaum ()` mit n Knoten, d.h. `size t == n`, erzeugt. / 3

Beispielsweise sollen für $n = 2$ die folgenden Bäume erzeugt werden:

```
(Node () (Node () Empty Empty) Empty)
(Node () Empty (Node () Empty Empty))
```

39. Blocksatz **6 Punkte**

Schreiben Sie eine Funktion `block :: Int -> String -> [String]`, die einen Text in Blocksatz setzt. Dabei wird die Anzahl der Zeichen pro Zeile als `Int`-Parameter vorgegeben. Sie können voraussetzen, dass der Text als Trennzeichen nur Leerzeichen und Zeilenumbruchzeichen (`'\n'`) enthält und dass kein Wort länger als die vorgegebene Zeilenlänge ist. Texte sollen mit der folgenden Funktion in Blocksatz ausgegeben werden.

```
printBlock   :: Int -> String -> IO ()
printBlock n =  putStr . unlines . (block n)
```

40. `foldr`-Eigenschaft **2 Punkte**

Zeigen Sie, dass für beliebige assoziative Funktionen `f :: a -> a -> a` mit neutralem Element `e :: a` und für beliebige endliche Listen `xs :: [a]` gilt:

```
foldr f e (xs++ys) = (foldr f e xs) `f` (foldr f e ys)
```