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

D-35032 Marburg
Hans Meerwein Straße
Lahnberge

21. Januar 2004

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

Nr. 11, Besprechung bzw. Abgabe: 28. und 29. Januar in den Übungen

---

Hinweis: Am **28. Januar** findet die Vorlesung ausnahmsweise im Hörsaal **HG 4** (direkt neben HG 5) statt.

A. Mündliche Aufgaben

51. Typinferenz bei überladenen Funktionen

Geben Sie zu jeder der nachfolgenden bedingten und unbedingten Gleichungen, die zusammen die Funktion `merge` definieren, den Typ an und leiten Sie hieraus den Typ der Funktion `merge` her.

```
merge (x:xs) (y:ys)
   | x<y        = x : merge xs (y:ys)
   | x==y       = x : merge xs ys
   | otherwise  = y : merge (x:xs) ys
merge (x:xs) [] = x : xs
merge [] (y:ys) = y : ys
merge [] []     = []
```

52. Strikte/nicht-strikte Auswertung

Zum Sortieren kann die Funktion `insertionSort` verwendet werden:

```
insertionSort :: Ord a => [a] -> [a]
insertionSort = foldr insert []

insert :: Ord a => a -> [a] -> [a]
insert x []     = [x]
insert x (y:ys) | x <= y    = x : y : ys
                | otherwise = y : insert x ys
```

Geben Sie für den Ausdruck `insertionSort [3,4,2,1]` eine

(a) Leftmost-Innermost-Auswertung und eine (b) Leftmost-Outermost-Auswertung

an. Zeigen Sie, wo sich die beiden Reduktionsstrategien voneinander unterscheiden.

B. Hausaufgaben

53. Gegeben sei eine Funktion `f :: [a] -> [b] -> a -> b`. $\boxed{\text{2 Punkte}}$

(a) Welchen Typ hat der Ausdruck `f [] []`? / 1

(b) Welchen Typ hat die Funktion `h` mit `h x = f x x`? / 1

Begründen Sie Ihre Antworten.

54. Eingabe von Bäumen **4 Punkte**

Ein *Parser* erhält als Eingabe eine Zeichenkette. Er überführt ein Anfangsstück seiner Eingabe in einen „semantischen“ Wert, der zusammen mit der restlichen Eingabe zurückgegeben wird. Als Typ eines Parsers wählt man den Typ `ReadS`, der wie folgt als Typsynonym vordefiniert ist: `type ReadS a = String -> [(a, String)]`

Der Ergebnistyp ist eine Liste, weil es mehrere Möglichkeiten geben kann, Teile der Eingabe zu parsen. Für die in Haskell vordefinierte Funktion `reads :: Read a => ReadS a` gilt etwa: `(reads :: ReadS Int) "10 Skripten" => [(10," Skripten")]`
Die gewünschte Typinstanz von `reads` kann aus dem Aufruf nicht ermittelt werden und muss daher explizit vorgegeben werden.

Für die Erkennung von Zeichenfolgen ist die Parserfunktion `lex :: ReadS String` vordefiniert. Es gilt beispielsweise:

```
lex "10 Skripten"           => [("10"," Skripten")]
lex "Skripten zu verkaufen" => [("Skripten"," zu verkaufen")]
```

Definieren Sie eine Funktion `readsTree :: (Read a) => ReadS (Tree a)` für Binärbäume des Typs
`data Tree a = Empty | Node a (Tree a) (Tree a) deriving Show`
Es soll gelten:

```
(readsTree :: ReadS (Tree Int)) "(Node 5 (Node 3 Empty Empty) Empty)"
 => [(Node 5 (Node 3 Empty Empty) Empty,"")]
```

55. Huffman-Kodierung **6 Punkte**

Die Huffman-Kodierung ist eine verlustlose Kompressionsmethode, bei der häufig vorkommende Buchstaben durch kürzere Bitfolgen dargestellt werden als seltene. Anhand eines Referenztextes werden die Häufigkeiten der darin vorkommenden Buchstaben bestimmt.

Die Kodierung erfolgt in den folgenden Schritten:

(a) Bestimmung der Buchstabenhäufigkeiten im Referenztext
(vgl. Funktion `frequency` aus Übung 8, Aufgabe 36).
Z.B.: `frequency "battat" -> [('b',1),('a',2),('t',3)]`

(b) Erzeugen des Huffman-Baums aus der Häufigkeitsliste
Z.B.: `Node 6 (Node 3 (Leaf 'b' 1) (Leaf 'a' 2)) (Leaf 't' 3)`

(c) Umwandeln des Huffman-Baums in eine Tabelle
Z.B.: `[('b',[L,L]),('a',[L,R]),('t',[R])]`

(d) Kodieren der Daten mit der erzeugten Tabelle
Z.B.: `"battat" -> [L,L,L,R,R,R,L,R,R]`, also 9 Bits statt 6 Bytes.

Die Dekodierung verläuft ähnlich:

(a) Erzeugen des Huffman-Baums aus dem selben Referenztext, der bei der Kodierung benutzt wurde

(b) Dekodieren der kodierten Daten durch Nachsehen im Huffman-Baum.

Benutzen Sie die auf der Internetseite der Vorlesung vorgegebenen Module `Types.hs` und `Frequency.hs`, um ein Modul `Coding` zu erstellen, das Kodier- und Dekodierfunktionen `code` und `decode` nach der Huffman-Methode bereitstellt.