Übungen zu Einführung in die Informatik I

Aufgabe 27 Termersetzung: Terminierung

In der Aufgabe 20, Übungsblatt 5, wurden die Termersetzungsregeln (1)-(3) zur Überführung eines Konstruktortermes in Normalform angegeben.

a) Geben Sie eine Bewertungsfunktion $g : \{t \in W_{\text{SEQ}} : t^{\textbf{SEQ}} \neq \bot\} \to \mathbb{N}$ an, die jedem Konstruktorterm $t \in W_{\text{SEQ}}$ ein Gewicht $g(t)$ zuordnet, so daß für die ersten drei Regeln `l` $\to$ `r` des Termersetzungssystem von Aufgabe 20 gilt: `g(l) > g(r)`.

b) Was läßt sich aus der Existenz dieser Bewertungsfunktion g für das Termersetzungssystem schließen. Begründen Sie Ihre Antwort.

Aufgabe 28 BNF

a) Beschreiben Sie mit BNF-Regeln die formale Sprache S_{ab}, die aus allen Wörtern besteht, in denen abwechselnd 'a' und 'b' auftreten (z. B. a, b, ab, ba, ...).

b) Allgemein werden durch BNF-Regeln formale Sprachen $X_j = \cup_{i \in \mathbb{N}} X_j^i$ definiert, wobei X_j die formale Sprache des regulären Ausdrucks R_j ist, für den das Nonterminal <e_j> als Abkürzung verwendet wird. Charakterisieren Sie für folgende BNF-Regel

<K> ::= {(<K>)}*

die formalen Sprachen X_j induktiv, indem Sie die Folge der regulären Ausdrücke R_j^0, R_j^1, R_j^2 und die zugehörigen formalen Sprachen X_j^0, X_j^1, X_j^2 angeben, wobei

$$\begin{array}{rcll} R_j^0 & ::= & \{\} & \text{reg. Ausdruck mit Sprache } \emptyset \\ R_j^{i+1} & ::= & R_j[R_1^i/\texttt{<}e_1\texttt{>}, \ldots, R_n^i/\texttt{<}e_n\texttt{>}]; & 1 \leq j \leq n \end{array}$$

c) Beschreiben Sie informell den Aufbau korrekt geklammerter boolescher Terme („korrekt geklammerte“ boolesche Terme sind nicht notwendigerweise vollständig geklammert).

d) Definieren Sie die formale Sprache der korrekt geklammerten booleschen Terme mittels BNF-Notation. Die Menge der Identifikatoren sei auf $ID = \{x, y, z\}$ beschränkt.

Aufgabe 29 **Beschreibung von HTML-Dokumenten mit BNF**

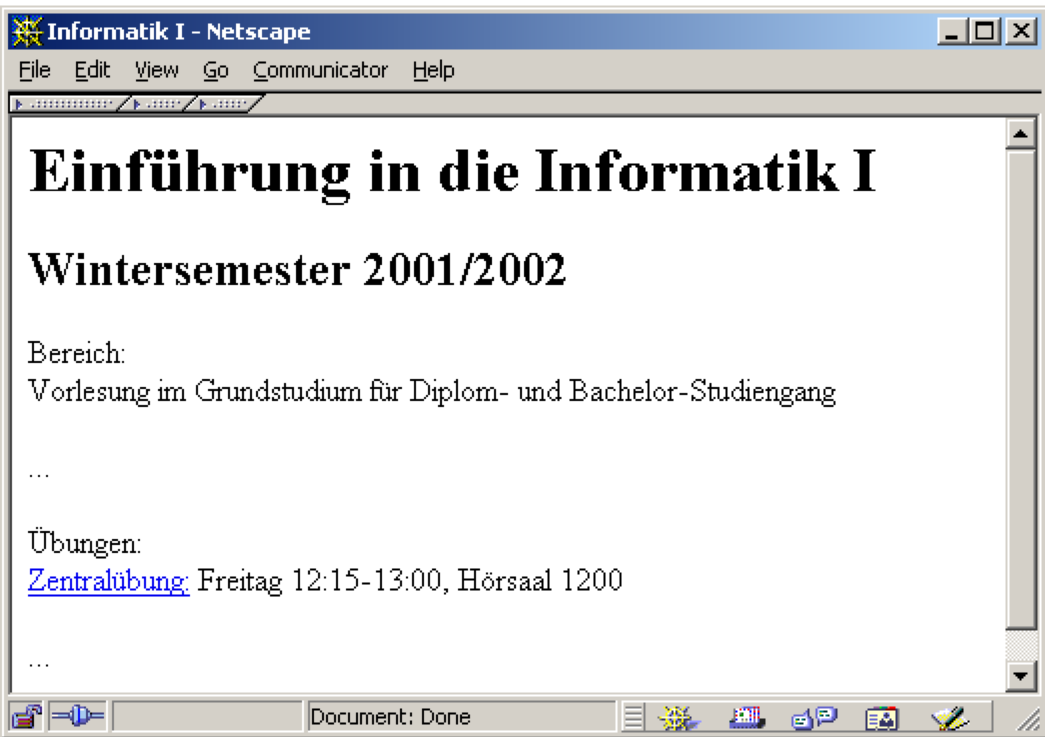
Ein HTML-Dokument enthält neben normalen Text auch Steuerelemente, die einen Browser (z.B. Netscape) informieren, wie bestimmte Teile des Dokuments behandelt werden sollen. Diese Steuerelemente nennt man Tags (engl. für Marke), mit denen z.B. festgelegt wird, ob ein Textabschnitt wie eine Überschrift, Absatz oder Hyperlink behandelt werden soll. Ein Tag wird dargestellt durch einen Elementnamen, der in spitzen Klammern eingeschlossen wird. HTML kennt zwei Formen von Steuerelementen bzw. Tags:

leere Tags	`<Elementname>`
und *Container*-Tags	`<Elementname>` *Teil des Dokuments* `</Elementname>`.

Im folgenden definieren wir einen Teilauschnitt von HTML als formale Sprache mit Hilfe der BNF-Notation:

```
<HTML-Dokument>     ::=  <html> <Dokumentinhalt> </html>
<Dokumentinhalt>    ::=  <head> <Kopf> </head>
                         <body> <Rumpf> </body>
<Kopf>              ::=  <title> <Text> </title>
<Rumpf>             ::=  { <Überschrift> | <Absatz> }*
<Überschrift>       ::=  <h1> { <HTMLtext> }* </h1> |
                         <h2> { <HTMLtext> }* </h2>
<Absatz>            ::=  <p> { <HTMLtext> }* </p>
<HTMLtext>          ::=  <Text> | <Hyperlink> | <br> <HTMLtext>
<Hyperlink>         ::=  <a href="<URL>"> <Text> </a>
<URL>               ::=  <Text>
<Text>              ::=  { A | B | C ... | Z | a | b | c ... | z |
                         Ä | Ö | Ü | ä | ö | ü | ß | 0 | 1 | 2 ... | 9 |
                         ␣ | \n | , | ; | . | : | ! | ? | / | ( | ) - | + }*
```

a) Leiten Sie mit Hilfe obiger BNF-Regeln ein HTML-Dokument mit dem Titel “Informatik I” ab, das von einem Browser wie folgt dargestellt wird:



b) Folgender Ausschnitt eines HTML-Dokuments wird in einem Browser als Aufzählung dargestellt. Erweitern Sie die gegebenen BNF-Regeln um das Aufzählungskonstrukt:

```
<html>
<head>
...
</head>
<body>
...
<p>
Inhalt:
<br>Die Vorlesung behandelt
</p>
<ul>
<li>Information und ihre Repräsentation</li>
<li>Rechenstrukturen und Algorithmen</li>
<li>Applikative Programmiersprachen</li>
<li>Zuweisungsorientierte Ablaufstrukturen</li>
<li>Sortendeklarationen</li>
<li>Maschinennahe Syntaxelemente</li>
<li>Rekursive Sortendeklarationen</li>
</ul>
...
</body>
</html>
```

Hinweis: Sie können in der Sunhalle eine private Hompage anlegen. Informationen dazu finden Sie unter http://www.in.tum.de/rechenbetrieb/infohalle/user-home.html .

Aufgabe 30 (P) Pattern matching

a) Implementieren Sie unter Verwendung von Pattern matching folgende Funktionen:

- `isEmpty :: [a] -> Bool`; ermittelt, ob eine Sequenz leer ist oder nicht.
- `rest :: [a] -> [a]`; gibt die Sequenz ohne deren erstes Element zurück; wird die Funktion für eine leere Sequenz aufgerufen, so ist das Ergebnis nicht definiert.

b) Durch

```
data List a = Empty | Append (a, List a)
```

ist der Datentyp einer Liste über Elementen vom Typ `a` definiert. Implementieren Sie unter Verwendung von Pattern matching folgende Funktionen:

- `istLeer :: List a -> Bool`; ermittelt, ob die Liste leer ist.
- `erstesElement :: List a -> a`; gibt das erste Element der Liste zurück; wird die Funktion für eine leere Sequenz aufgerufen, so ist das Ergebnis nicht definiert.
- `verbinde :: (List a, List a) -> List a`; hängt zwei Listen aneinander.

c) Das Datum ist durch die Angabe von drei ganzen Zahlen (Tag , Monat, Jahr) gegeben. Definieren Sie einen Datentyp namens `Datum` (kein Typsynomym !), der diese Informationen enthält. Implementieren Sie anschließend eine Funktion namens `nextDay`, die zu einem gegebenen Datum das Datum des nächsten Tages zurückgibt! Beachten Sie dabei, dass die Monate unterschiedlich viele Tage haben, dass es Schaltjahre gibt, wobei der Julianische Kalender (d.h. jedes Jahr, dessen Jahreszahl durch vier teilbar ist, ist ein Schaltjahr) zugrundegelegt werden kann. (**Hinweis:** Bei dieser Aufgabe empfiehlt es sich ausgiebig von Hilfsfunktionen Gebrauch zu machen!)