Lehrstuhl für Programmiersprachen und Übersetzerkonstruktion
CAU Kiel, Institut für Informatik
Prof. Dr. M. Hanus

24118 Kiel
Christan-Albrechts-Platz 4
Tel.: 0431 / 880-7271

2. Übung „Übersetzerbau“

Bearbeitung bis zum 29. April 2008

Bitte senden Sie alle Programmieraufgaben zusätzlich zur Abgabe in ausgedruckter Form auch per Email an Axel Stronzik (`axs@informatik.uni-kiel.de`)!

Aufgabe 4
In dieser Aufgabe sollen Funktionen für binäre Suchbäume (ohne Höhenbalancierung) in Haskell implementiert werden.
Definieren Sie für die Darstellung binärer Suchbäume eine Datenstruktur `SearchTree`, die nur in den inneren Knoten Werte erlaubt.
Programmieren Sie folgende Funktionen für das Arbeiten mit binären Suchbäumen. Die Elemente des Baums sollen hierbei einen Typ der Klasse `Ord` haben (diese Klasse wird in der Übung am 23.4. vorgestellt).

- `insertST` fügt eine Wert unter Berücksichtigung der Ordnung in einen Suchbaum ein und liefert den veränderten Suchbaum als Ergebnis.
- `listToST` wandelt eine Liste in einen Suchbaum um (die Elemente werden hierbei sortiert).
- `sTToList` wandelt einen Suchbaum in eine sortierte Liste um.
- `deleteST` löscht ein Element in einem Suchbaum und liefert den veränderten Suchbaum.
- `mapST` und `foldST`, welche analog zu `map` bzw. `foldr` für binäre Suchbäume definiert sind. Beachten Sie, dass beim “natürlichen” Falten die Typen der Funktionsparameter den Typen der Konstruktoren entsprechen sollten.

  Definieren Sie die Funktion `listToST` unter Verwendung von `foldST` neu.

Geben Sie wie immer auch die Typen sämtlicher definierten Funktionen an.

Aufgabe 5
In der Vorlesung wurde die Programmiersprache Simple und Haskell-Datenstrukturen zur abstrakten Repräsentation von Simple-Programmen vorgestellt, vgl.

`http://www.informatik.uni-kiel.de/~mh/lehre/cb08/simpleprog.hs`

Definieren Sie einen Interpreter für Simple-Programme, der alle Ausgaben des Simple-Programms in einer Liste von Zahlen aufsammelt, also eine Funktion

```
simpleInt :: Stm -> [Int]
```

Hierzu sollte Sie eine Variablenbelegung als Liste von Tupeln von Bezeichnern und Werten darstellen:

```
type Bindings = [(Id,Int)]
```

Definieren Sie zunächst eine Funktion

```
getBind :: Id -> Bindings -> Int
```

welche die Bindung einer Variablen bestimmt. Definieren Sie außerdem eine Funktion

```
updateBind :: Id -> Int -> Bindings -> Bindings
```

zum Aktualisieren von Variablenbindungen.
Unter Verwendung von Variablenbelegungen können Sie dann eine Hilfsfunktion

```
stmCalc :: Bindings -> Stm -> (Bindings,[Int])
```

definieren, welche die Variablenbelegungen berechnet und gleichzeitig die Ausgabe aufbaut. Der Zugriff auf eine nicht belegte Variable soll einen Laufzeitfehler ergeben (hierzu ist die Haskell-Funktion `error :: String -> a` ganz nützlich).
Überlegen Sie auch, wie Sie eine entsprechende Funktion für Ausdrücke definieren können. Beachten Sie hierbei, dass innerhalb von Ausdrücken auch wieder Anweisungen und damit Ausgaben möglich sind.

Aufgabe 6
Sei PEVAL ein partieller Auswerter geschrieben in der Sprache I. Zeigen Sie, dass

$$cc := I\ \mathsf{PEVAL}(\mathsf{PEVAL}, \mathsf{PEVAL})$$

ein Compiler-Compiler ist, d.h. einen Interpreter in einen Compiler übersetzt.