Lehrstuhl für Programmiersprachen und Übersetzerkonstruktion
CAU Kiel, Informatik und Prakt. Mathematik
Prof. Dr. M. Hanus, Dr. F. Huch

24118 Kiel
Christan-Albrechts-Platz 4
Tel.: 0431 / 880-7277

2. Übung „Übersetzerbau“

Abgabe bis zum 29. Oktober, 12 Uhr in Raum 704

Die Abgabe dieses Übungsblattes sollte bis Mittwoch mittag erfolgen. Danach werden wir dann auf die Montagsabgabe umstellen. Alle Programmieraufgaben sollten Sie zusätzlich zur Abgabe eines Ausdrucks auch per EMail an Hauke Tschach (`hts@informatik.uni-kiel.de`) schicken.

Aufgabe 4 10 Punkte

In dieser Aufgabe sollen Funktionen für binäre Suchbäume (ohne Höhenbalancierung) in Haskell implementiert werden.
Definieren Sie für die Darstellung binärer Suchbäume eine Datenstruktur `Baum`, die sowohl in den Blättern, als auch in den 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.

- `einfuegen` fügt eine Wert unter Berücksichtigung der Ordnung in einen Suchbaum ein und liefert den veränderten Suchbaum als Ergebnis.
- `baum` wandelt eine Liste in einen Suchbaum um (die Elemente werden hierbei sortiert) .
- `liste` wandelt einen Suchbaum in eine sortierte Liste um.
- `loeschen` löscht ein Element in einem Suchbaum und liefert den veränderten Suchbaum.
- `mapbaum` und `foldbaum`, welche analog zu `map` bzw. `foldr/foldl` für binäre Suchbäume definiert sind.

  Definieren Sie die Funktion `liste` unter Verwendung von `foldbaum` neu.

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

Aufgabe 5 10 Punkte

In der Vorlesung wurde die Programmiersprache Simple und Haskell-Datenstrukturen zur abstrakten Repräsentation von Simple-Programmen vorgestellt.
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]
```

Wir definieren hierzu eine Variablenbelegung als Liste von Tupeln von Bezeichnern und Werten:

```
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.
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.