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

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

1. Übung „Übersetzerbau“

Bearbeitung bis zum 25. April 2006

In dieser ersten Übung sollen Sie sich mit dem Hugs-System vertraut machen.
Public Domain Versionen und weitere Informationen zur Vorlesung und den Übungen finden Sie im WWW unter der Adresse:

```
http://www-ps.informatik.uni-kiel.de/~fhu/CBS06/
```

Aufgabe 1 2 Punkte

Die Funktion `double` sei wie folgt definiert:

```
double x = x + x
```

Geben Sie alle möglichen Auswertungen von `double (double 3)` an.

Aufgabe 2 1+3 Punkte

In der Haskell-Einführung wurde die Fibonacci-Funktion intuitiv und mittels Akkumulatortechnik definiert.
Berechnen Sie `fib1`/`fib2` `5` und `fib1`/`fib2` `30` und geben Sie die Zahl der benötigten Reduktionsschritte (entspr. Zeitbedarf) und der benötigten Zellen (entspr. Platzbedarf) an. Hugs gibt beides nach Eingabe der Anweisung '`:set +s`' aus.
Bestimmen Sie anhand der benötigten Reduktionsschritte die Zeitkomplexitäten der Implementierungen (`fib1`/`fib2` liegt in $O(??)$). Die einfachste Methode ist eine graphische Auswertung.

Aufgabe 3 2+1+1 Punkte

Programmieren Sie folgende Funktionen in Haskell und geben Sie jeweils auch den Typ der definierten Funktionen an:

a) Definieren Sie eine Funktion `rev`, welche eine Liste umdreht. Welchen Aufwand hat Ihr Algorithmus?

   Optimieren Sie Ihre Lösung mit Hilfe der Akkumulatortechnik auf lineare Komplexität (mit Begründung).

b) Definieren Sie eine zweistellige Funktion `stelle`, die feststellt, an welcher Position ein Element in einer Liste vorkommt. Falls das Element nicht in der Liste vorkommt, soll die Funktion den Wert 0 liefern.

   Beispiel: `stelle 1 [2,1,3,1]` ergibt `2`

c) Definieren Sie eine Funktion `inconcat`, die eine Liste von Strings zu einem neuen String konkateniert und dabei zwischen je zwei Elemente eine Zeichenkette einfügt.

Beispiel: `inconcat "-" ["Christian","Albrechts","Universität"]`
ergibt `"Christian-Albrechts-Universität"`

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 `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. Sollten Sie diese Aufgabe bearbeiten, bevor Sie die Klasse `Ord` kennen gelernt haben, verwenden Sie zunächst Suchbäume, die Elemente vom Typ `Int` enthalten.

- `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 Konstruktuoren entsprechen sollten.

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

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

Wichtige Interpreter-Kommandos des Hugs-Systems:

- `:l` *filename*

  Löschen aller vorhandenen Funktionsdefinitionen (außer Prelude) und Laden des Programms *filename*.

- `:r`

  Wiederholt die letzte Ladeoperation, somit werden alle Funktionsdefinitionen aktualisiert.

- `:q`

  Verlassen des Hugs-Systems.

- `:?`

  Komandoliste wird angezeigt.

- *Ausdruck* (z.B. `fac 3`)

  Start einer Funktionsberechnung.