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

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

Übungsaufgaben zum Praktikum „Internet-Programmierung“

Hier sind einige Übungsaufgaben zur Einarbeitung in Curry. Bearbeitung von Aufgabe 1 und 2 am besten heute nach dem Curry-Kurs in Raum 709. Sonst Besprechung am Montag nach dem Kurs und dann Bearbeitung von Aufgabe 3. Ggf. auch schon Vorbereiten von Aufgabe 3 bis Montag.
Geben Sie immer die Typen der definierten Funktionen an!

Aufgabe 1
Definieren Sie die Fibonacci-Funktion rekursiv.
Berechnen Sie `fib 5`, `fib 10` und `fib 20` und geben Sie die Zeit an, die die Berechnung benötigt (`:set` in PAKCS verwenden).
Schätzen Sie den Aufwand der Funktion ab. Entwickeln Sie eine effizientere Implementierung mit linearer Laufzeit.

Aufgabe 2
Programmieren Sie folgende Funktionen in Curry und geben Sie jeweils auch den Typ der definierten Funktionen an:

a) Definieren Sie eine zweistellige Funktion `position`, 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: `position 1 [2,1,3,1]` ergibt `2`

b) 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"`

c) Definieren Sie eine Funktion `scanTags`, die aus einem String alle Tags extrahiert.
Beispiel: `scanTags "<a>Link</a> <p>Kein Link"` ergibt `["a","/a","p"]`

d) Definieren Sie eine zweistellige Funktion `checkTags`, die überprüft, ob nur bestimmte Tags in einem String vorkommen.
Beispiel:
`checkTags ["a","p"] "<a>Link</a> <p>Kein Link"` ergibt `False`
`checkTags ["a","/a","p"] "<a>Link</a> <p>Kein Link"` ergibt `True`

e) Definieren Sie eine Funktion `permute :: [a] -> [[a]]`, die die Liste aller Permutationen einer Liste berechnet.
Beispiel:
`permute [1,2,3]` ergibt
`[[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]`
oder eine List mit den gleichen Elementen, aber in anderer Reihenfolge.

Aufgabe 3
In dieser Aufgabe sollen Funktionen für binäre Suchbäume (ohne Höhenbalancierung) in Curry implementiert werden.
Definieren Sie für die Darstellung binärer Suchbäume eine Datenstruktur `Tree`, die sowohl in den Blättern, als auch in den Knoten Werte erlaubt.
Programmieren Sie folgende Funktionen für das Arbeiten mit binären Suchbäumen.

- `insert` fügt eine Wert unter Berücksichtigung der Ordnung in einen Suchbaum ein und liefert den veränderten Suchbaum als Ergebnis.
- `listToTree` wandelt eine Liste in einen Suchbaum um (die Elemente werden hierbei sortiert) .

  Definieren Sie die Funktion `listToTree` unter Verwendung von `foldr`.
- `treeToList` wandelt einen Suchbaum in eine sortierte Liste um.
- `delete` löscht ein Element in einem Suchbaum und liefert den veränderten Suchbaum.
- `mapTree` und `foldTree`, welche analog zu `map` bzw. `foldr` für binäre Suchbäume definiert sind.

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