P. Thiemann, M. Neubauer Wintersemester 2004/2005

Praktikum Compilerbau

http://proglang.informatik.uni-freiburg.de/teaching/compilerbau/2004/

Übungsblatt 1

Abgabe: 10.11.2004

Aufgabe 1 (Funktionen über Listen):

- Implementiere die Funktion `c2f_list : float list -> float list`, die eine Liste von Temperaturen von "Celsius" in "Fahrenheit" konvertiert.
- Implementiere eine Funktion, die aus einer Liste alle Elemente, die grösser als k sind entfernt.
- Implementiere eine Funktion, die ein durch eine List von Koeffizienten repräsentiertes Polynom, mit Hilfe des Horner-Schemas an einer bestimmten Stelle auswertet.

Aufgabe 2 (Funktionen höherer Ordnung):

- Implementiere die Funktion `c2f_list : float list -> float list`, die eine Liste von Temperaturen von "Celsius" nach "Fahrenheit" konvertiert unter Verwendung von `List.map`.
- Implementiere eine Funktion, die alle Funktionen in einer Liste miteinander komponiert unter Verwendung von `List.fold_right`.
- Implementiere `map` unter Verwendung von `List.fold_right`.
- Implementiere eine Funktion, die ein durch eine List von Koeffizienten repräsentiertes Polynom, mit Hilfe des Horner-Schemas an einer bestimmten Stelle auswertet unter Verwendung von `List.fold_right`.
- Implementiere eine Funktion, die aus eine Liste alle Elemente, die grösser als k sind, entfernt unter Verwendung von `List.filter`.
- Schreibe eine Funktion, die die folgende Signatur besitzt:
  `cb : (('a -> unit) -> unit) -> ('a -> ('b ->  unit) -> unit) -> ('b -> unit) -> unit.`

Aufgabe 3 (Datentypen):

- Implementiere eine Funktion, die zwei Vektoren (repräsentiert als Liste von Zahlen) komponentenweise mit Hilfe von `map`, `zip` und `uncurry` mulitpliziert.
- Implementiere eine Funktion `mirror : 'a ltree ltree -> 'a ltree`, die Bäume spiegelt.
- Implementiere einen Interpreter, der symbolische Ausdrücke mit Variablen unter einer einer Variablenbelegung auswertet.

- Definiere den Datentyp und zumindest die Einfügeoperation für balancierte Bäume (AVL oder Red-Black).
- Implementiere eine iterative Variante der Fibonacci-Funktion mit Hilfe von Referenzen.

Aufgabe 4 (Module und Strukturen):

- Erzeuge Strukturen `OrderedInt` sowie `IntSet`.
- Schreibe eine möglichst abstrakte Signatur `STRINGSET`, so dass `StringSet : STRINGSET` gilt.
- Schreibe eine Anwendung auf der Basis einer Struktur vom Typ `STRINGSET`. Zum Beispiel Einlesen einer Liste von Worten von einer Datei (ein Wort pro Zeile) und dann Test, ob das Wort auf der Kommandozeile in dieser Menge enthalten ist.
- Schreibe eine alternative Implementierung `FastSet` des Funktors von `ORDERED_TYPE`, so dass dessen Anwendung auf `tringSet` wieder eine Struktur vom Typ `STRINGSET` liefert. Kode und Funktionalität der Anwendung sollte unverändert bleiben.

Aufgabe 5 (SAX-Parser on OCaml):

Implementiere eine Version des SAX-Parsers von Übungsblatt 2, Aufgabe 4 der Compilerbau-Vorlesung in OCaml. Verwende als Scannergenerator Ocamllex oder Ulex[1].

[1] `http://www.cduce.org/download.html`