Hochschule München
Fakultät für Informatik und Mathematik
M. Ruckert

Theoretische Informatik II
Übungen

Listen in Scheme bestehen aus Paaren mit `car` und `cdr`, die man auch dazu verwenden kann binäre Bäume aufzubauen. In diesem Übungsblatt betrachten wir sortierte binäre Bäume. Jeder Knoten im Baum enthält dabei eine Zahl so dass alle Zahlen im linken Teilbaum kleiner und alle Zahlen im rechten Teilbaum größer sind. Für die Darstellung eines Knotens verwenden wir eine unechte (*improper*) Liste: `(zahl links . rechts)` . Ist `node` ein Knoten im Baum, so ist also `(car node)` die Zahl, `(cadr node)` der linke Teilbaum und `(cddr node)` der rechte Teilbaum. Der leere Baum wird durch eine leere Liste repräsentiert. Wir üben in diesem Übungsblatt die "destruktive" Veränderung dieser Datenstruktur.

Vorbereitung: Schreiben Sie folgende Getter, Setter und Test Funktionen: `value`, `left`, `right`, `value!`, `left!`, `right!`, `empty?`,

Aufgabe 15. Schreiben Sie eine Funktionen `(insert tree n)`, die die Zahl `n` destruktiv in den Baum `tree` einfügt und den veränderten Baum zurück gibt.

Hinweis: In einer Sprache wie Scheme, die nur Pass-by-Value aber keinen Pass-by-Reference kennt, kann man bei Funktionen wie `(insert tree n)` eine Variable `tree` innerhalb der Funktion nicht verändern, da nur der Wert übergeben wird. Man umgeht dieses Problem indem man die Zuweisung der aufrufenden Funktion überläßt. Der Aufruf hat dann immer die Form: `(set! tree (insert tree n))`.

Aufgabe 16. Schreiben Sie eine Funktionen `(print_tree t)`, die den Baum `t` in einer lesbaren "Baumform" ausgibt. Am einfachsten läßt man dabei den Baum von links nach rechts wachsen. D.h. zuerst kommt der rechte Teilbaum, der etwas stärker eingerückt ist als der Wurzelknoten, dann der Wert des Wurzelknoten, dann der linke Teilbaum, ebenfalls eingerückt. Das könnte dann so aussehen:

```
    6
      5
4
      3
   2
      1
```

Aufgabe 17. Schreiben Sie eine Funktionen `(smallest t)`, die die kleinste Zahl `n` zurück gibt (oder `#f` falls der Baum leer ist) und `(remove-smallest t)`, die sie aus dem Baum löscht.

In Scheme heissen die Arrays "Vektoren". Da wir etwas Umgang damit für den Scheme Interpreter brauchen hier einige Übungen im "prozeduralen Programmieren".

Aufgabe 18. Schreiben sie eine Funktion `(qsort v)`, die einen Vector `v` mittels quicksort (oder einem anderen Sortierverfahren Ihrer Wahl) sortiert.

Aufgabe 19. Schreiben sie eine Funktion `(qsort v less)`, die einen Vector `v` mittels quicksort (oder ...) sortiert und zum Vergleich zweier Elemente die Funktion `less` verwendet.