X. Übungsblatt "Informatik I" WS 95/96

Christian Schulte, Prof. Gert Smolka

Ausgabe: 22. Dezember 1995

Abgabe: 12. Januar 1996 vor der Vorlesung (Programme per E-Mail)

Aufgabe X.1 (Dictionaries (5)) Ein *Dictionary* ist eine Datenstruktur, mit der durch Schlüssel identifizierte Werte abgelegt und abgefragt werden können. Dabei kann zu einem Schlüssel höchstens ein Wert abgelegt werden. Im folgenden nehmen wir an, daß Schlüssel Symbole sind. Schreiben Sie eine Klasse `(dictionary)`, die die Einträge als eine nach den Schlüsseln sortierte Liste verwaltet. Ein Dictionary `d` soll die folgenden Methoden haben:

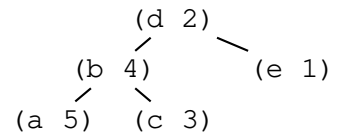
- `((d 'put) s x)` speichert `x` unter dem Schlüssel `s`.
- `((d 'get) s)` evaluiert zu dem unter dem Schlüssel `s` gespeicherten Wert.
- `((d 'remove) s)` entfernt, falls vorhanden, einen unter dem Schlüssel `s` gespeicherten Wert.
- `((d 'in?) s)` testet, ob unter dem Schlüssel `s` ein Wert gespeichert ist.

Hinweis: In Scheme kann mit der folgenden Prozedur getestet werden, ob das Symbol `a` lexikalisch kleiner als das Symbol `b` ist:

```
(define (symbol<? a b)
   (string<? (symbol->string a) (symbol->string b)))
```

So evaluiert `(symbol<? 'a 'b)` zu `#t`, `(symbol<? 'b 'a)` zu `()`, `(symbol<? 'aba 'abb)` zu `#t` und `(symbol<? 'ba 'b)` zu `()`.

Aufgabe X.2 (Binäre Suchbäume (5)) In dieser Aufgabe lernen Sie eine Datenstruktur kennen, mit der sich Dictionaries effizienter implementieren lassen. Jeder Knoten eines *binären Suchbaumes* hat einen Eintrag aus Schlüssel (ein Symbol) und Wert. Dabei gilt, daß der Schlüssel größer ist als alle Schlüssel im linken Unterbaum und kleiner als alle Schlüssel im rechten Unterbaum. Zusätzlich gibt es noch den leeren binären Suchbaum.
Graphisch kann man einen binären Suchbaum so darstellen:



1. Zeichen Sie alle möglichen binären Suchbäume, die alle Einträge aus dem obigen Beispiel enthalten.
2. Schreiben Sie eine Prozedur `(fuege-ein b s x)`, die in einen binären Suchbaum `b` unter dem Schlüssel `s` den Wert `x` einträgt. Dabei soll der Baum `b` verändert werden (mit Hilfe der primitiven Prozeduren `set!` und/oder `set-car!` und/oder `set-cdr!`).
3. Implementieren Sie Dictionaries (wie in Aufgabe X.1) mit binären Suchbäumen.

Aufgabe X.3 (Priority-Queues (5)) In der Vorlesung haben Sie bereits gelernt, was eine Queue ist. Eine *Priority-Queue* erlaubt es, Elemente beim Einfügen zusätzlich mit einer Priorität (einer natürlichen Zahl) zu versehen. Beim Entfernen aus einer Priority-Queue wird zuerst das älteste Element mit maximaler Priorität entfernt.
Schreiben Sie eine Klasse `(prio-queue)`, die eine Priority-Queue erzeugt. Eine Priority-Queue `q` soll die folgenden Methoden haben:

- `((q 'null?))` testet, ob `q` leer ist.
- `((q 'dequeue))` evaluiert zu dem ältesten Element mit maximaler Priorität, und entfernt dieses Element.
- `((q 'enqueue) p x)` fügt `x` mit Priorität `p` (eine natürliche Zahl) ein.

Hinweis: Implementieren Sie Priority-Queues durch geordnete Listen von priorisierten Einträgen. Verändern Sie diese Liste beim Einfügen von Elementen so, daß nur eine Cons-Zelle erzeugt wird. Dabei ist es hilfreich, wenn die Liste stets ein "Dummy"-Element mit negativer Priorität enthält, um Probleme mit der leeren Liste zu vermeiden.

Aufgabe X.4 (Semantikbäume für Scheme 2 (5)) Zu welchem Wert und Zustand evaluieren die beiden folgenden Ausdrücke in der leeren Umgebung und dem leeren Zustand:

1.
```
new x in
    x := (cons (quote a) x) ;
    if (eq? x (cdr x)) then (set-car! x (cdr x)); (eq? x (car x))
    else 4 end
end
```

2.
```
new x in new y in
    x := lambda (y) (set-car! y x) end;
    y := (cons 7 (x y));
    if (eq? x (car y)) then 42 else 4 end
end end
```

Geben Sie zu beiden Ausdrücken einen Semantikbaum an, der nur die angewendeten Regeln angibt.

Zusatzaufgabe X.5 (Zirkuläre Listen (10)) Schreiben Sie eine Prozedur, die testet, ob eine Liste zirkulär ist. Greifen Sie dabei die folgende Idee auf:

> Zwei Läufer, von denen einer doppelt so schnell läuft wie der andere, starten auf einer kreisförmigen Rennbahn von der gleichen Stelle in die gleiche Richtung. Interessant ist, daß nach einer gewissen Zeit beide Läufer gleichauf sind. Würden die beiden Läufer auf einer geraden Rennbahn mit einer gewissen Länge laufen, so würden sie niemals gleichauf sein.

Argumentieren Sie, wieso Ihre Prozedur funktioniert.

Dieses Übungsblatt ist das erste, dessen Punkte für die zweite Hälfte der Vorlesung zählen.

Wir wünschen Ihnen frohe Weihnachten und einen guten Rutsch ins neue Jahr!