



Institut für Informatik
Prof. Dr. Peter Thiemann
Jochen Walter

Georges-Köhler-Allee 79
D-79110 Freiburg i. Br.

Freiburg, den 23. November 2000

Informatik 1, WiSe 2000/2001

Übungsblatt 4

Die Aufgaben werden in den Übungs- und Programmiergruppen
vom 23.11 bis zum 29.11. besprochen.

Die Lösungen müssen nicht abgegeben werden!

Die Aufgaben auf diesem Blatt können in Teams bearbeitet werden. Bevor drscheme gestartet werden kann, muß setup lang eingegeben werden. Der Sprachumfang sollte auf „Advanced Student“ eingestellt werden. Die Aufgaben sind mit einem bis drei Sternen versehen, wobei die Aufgaben mit einem Stern am einfachsten, die mit dreien am schwierigsten sind.

Aufgabe 1 (**):

Vorbemerkung: Um diese Aufgabe lösen zu können, muß man wissen, wie man in Scheme Zeichenketten (strings) angibt. Dies geschieht dadurch, daß man die Zeichen, aus denen sich die Zeichenkette zusammensetzt, in Anführungszeichen setzt. Zum Beispiel ist "Scheme" eine Zeichenkette, die sich aus den sechs Zeichen S, c, h, e, m und e zusammensetzt. Für manche Zeichen müssen besondere Maßnahmen ergriffen werden, wenn man sie innerhalb einer Zeichenkette hinschreiben möchte, näheres steht u.a. im Scheme-Report.

Gegeben sei folgender Scheme-Code:

```
;;; Datentyp "student"  
;;; Ein "student" ist eine Struktur (make-student vorname nachname matrikelnr)  
;;; wobei vorname eine Zeichenkette ist, die den Vornamen des Studenten  
;;; enthält, nachname eine Zeichenkette ist, die den nachnamen des Studenten  
;;; enthält und matrikelnr eine Zeichenkette ist, die die matrikelnummer  
;;; des Studenten enthält.  
(define-struct student (vorname nachname matrikelnr))
```

- Welche Funktionen werden durch diese Strukturdefinition von DrScheme automatisch zur Verfügung gestellt? Was bewirken sie?
- Wie erzeugen Sie einen Eintrag für einen Studenten namens Reinhard Franck mit der Matrikelnummer 0070815?
- Warum wird die Matrikelnummer durch eine Zeichenkette dargestellt und nicht durch eine Zahl?
- Reinhard Franck nimmt nach seiner Hochzeit den Namen seiner Frau Julie Reinhardson an. Schreiben Sie eine Funktion, die als Argumente eine student-Struktur und

einen Nachnamen verlangt und eine Kopie der `student`-Struktur zurückliefert, bei der das Nachnamenfeld durch den gesondert übergebenen Nachnamen ersetzt ist.

Lösung zu Aufgabe 1:

- (a) `student-make` erzeugt eine `student`-Struktur. Man muß beim Aufruf die Werte angeben, die die `vorname-`, `nachname-` und `matrikelnr-`Felder haben sollen.
`student-vorname` gibt den Wert des `vorname-`Feldes der übergebenen `student`-Struktur zurück.
`student-nachname` gibt den Wert des `nachname-`Feldes der übergebenen `student`-Struktur zurück.
`student-matrikelnr` gibt den Wert des `matrikelnr-`Feldes der übergebenen `student`-Struktur zurück.
`student?` gibt `true` zurück, falls der übergebene Wert eine `student`-Struktur ist, `false` sonst.
Daneben werden noch andere Funktionen erzeugt, die in der Vorlesung nicht behandelt wurden und deshalb auch nicht verwendet werden sollen.
- (b) `(load "ub4-1a.scm")`

```
(make-student "Reinhard" "Franck" "0070815")
```

- (c) Wie die vorige Teilaufgabe zeigt, möchte man Matrikelnummern mit führenden Nullen verwenden. Dafür ist die Darstellung durch Zahlen ungeeignet.
- (d) `(load "ub4-1b.scm")`

```
;;; SIGNATUR
;;; student-ersetze-nachname: student string -> student
;;; ERKLÄRUNG
;;; (student-ersetze-nachname s n) erzeugt eine Kopie von s, bei der das
;;; nachname-Feld durch n ersetzt ist.
;;; BEISPIEL
;;; (student-ersetze-nachname (make-student "Reinhard" "Franck" "0070815")
;;;                          "Reinhardson")
;;; DEFINITION
(define student-ersetze-nachname
  (lambda (s n)
    (make-student (student-vorname s)
                  n
                  (student-matrikelnr s))))
```

Aufgabe 2 (***):

Der folgende Scheme-Code

```
(define list-sum
  (lambda (liste)
```

```
(if (= (length liste) 0)
    0
    (+ (first liste) (list-sum (rest liste))))))
```

definiert eine Funktion, die als Argument eine Liste von Zahlen erwartet.

- Was berechnet diese Funktion?
- Beweisen Sie Ihre Antwort mittels vollständiger Induktion.
- Geben Sie die vollständige Funktionsspezifikation im Stile der in der Vorlesung gezeigten Funktionen an.

Lösung zu Aufgabe 2:

- Die Funktion berechnet die Summe der in der Liste enthaltenen Zahlen. Im Falle der leeren Liste wird 0 zurückgegeben: Für $\text{liste} = (a_1, a_2, \dots, a_n)$ gilt $(\text{list-sum liste}) = \sum_{i=1}^n a_i$.

- Induktionsanfang:** Für die leere Liste gilt die Aussage nach der Substitutionssemantik: $(\text{list-sum empty}) \rightarrow (\text{if} (= (\text{length empty}) 0) \dots) \rightarrow 0 \equiv \sum_{i=1}^0 a_i$

Induktionsschritt: Falls die Aussage für Listen der Länge n gilt, gilt sie auch für Listen der Länge $n + 1$:

```
(list-sum (cons a_0 liste)) →
(if (= (length (cons a_0 liste)) 0) ...) →
(+ (first (cons a_0 liste)) (list-sum (rest (cons a_0 liste)))) ≡
(+ a_0 (list-sum liste)) =
a_0 + ∑_{i=1}^n a_i ≡ ∑_{i=0}^n a_i ≡ ∑_{i=1}^{n+1} a_i
```

Dabei bedeutet \rightarrow einen Ableitungsschritt nach der Substitutionssemantik, \equiv eine algebraische Umformung und $=$ Verwendung der Induktionsannahme.

- ;;; SIGNATUR
 ;;; list-sum: list(number) -> number
 ;;; ERKLÄRUNG
 ;;; (list-sum liste) berechnet die Summe der in der Liste enthaltenen
 ;;; Zahlen. Falls liste die leere Liste ist, wird 0 zurückgegeben.
 ;;; BEISPIEL
 ;;; (list-sum (list 1 2 3))
 ;;; => 6

Aufgabe 3 (**):

Was ist der Unterschied zwischen einer iterativen – und einer rekursiven Berechnung?

Lösung zu Aufgabe 3:

Eine iterative Berechnung ergibt einen Berechnungsprozeß von konstanter Größe, bei einer rekursiven Berechnung ist das nicht der Fall.

Eine ausführlichere Darstellung dieses Sachverhalts findet man in Abschnitt 1.2 von „Structure and Interpretation of Computer Programs“

Aufgabe 4 (**):

Die Fibonacci-Zahlen F_i sind eine Zahlenfolge, die durch folgendes Bildungsgesetz bestimmt ist:

$$\begin{aligned}F_0 &:= 1 \\F_1 &:= 1 \\F_n &:= F_{n-1} + F_{n-2} \quad \text{für } n \geq 2\end{aligned}$$

Wandeln Sie folgende rekursive Funktion zur Berechnung der n -ten Fibonacci-Zahl so um, daß sie iterativ berechnet wird. Benutzen Sie dazu zwei Hilfsparameter, die die Zwischenergebnisse aufnehmen.

```
;;; SIGNATUR
;;; fibonacci-rec: number -> number
;;; ERKLÄRUNG
;;; (fibonacci-it n) berechnet die n-te Fibonacci-Zahl
;;; BEISPIEL
;;; (fibonacci-rec 3)
;;; => 3
;;; DEFINITION
(define fibonacci-rec
  (lambda (n)
    (if (< n 2)
        1
        (+ (fibonacci-rec (- n 1))
            (fibonacci-rec (- n 2)))))))
```

Lösung zu Aufgabe 4:

```
;;; SIGNATUR
;;; fibonacci-it: number -> number
;;; ERKLÄRUNG
;;; (fibonacci-it n) berechnet die n-te Fibonacci-Zahl
;;; BEISPIEL
;;; (fibonacci-it 3)
;;; => 3
;;; DEFINITION
(define fibonacci-it
  (lambda (n)
    (fibonacci-it-1 n 1 1)))
```

```
;;; SIGNATUR
;;; fibonacci-it-1: number number number -> number
;;; ERKLÄRUNG
;;; (fibonacci-it n p q) berechnet die n-te Zahl der Zahlenfolge mit dem
```

```

;;; Bildungsgesetz  $F_i = F_{i-1} + F_{i-2}$  und den Startwerten
;;;  $F_0 = p$  und  $F_1 = q$ 
;;; BEISPIEL
;;; (fibonacci-it-1 2 10 20)
;;; => 30
;;; DEFINITION
(define fibonacci-it-1
  (lambda (n p q)
    (cond
      ((= n 0)
       p)
      ((= n 1)
       q)
      (else
       (fibonacci-it-1 (- n 1) q (+ p q))))))

```

Aufgabe 5 (*):

Worin besteht der Unterschied zwischen diesem

```
(define number 5)
```

```
(* number number)
```

und jenem

```
(let ((number 5))
  (* number number))
```

Scheme-Code?

Lösung zu Aufgabe 5:

In beiden Fällen wird dem Wert 5 der Name `number` zugewiesen.

Lebensdauer: Im ersten Fall bleibt der Name auf Dauer erhalten, im zweiten Fall kann man den Namen nur während der Ausführung des Körpers der Let-Klausel verwenden. Das kann man überprüfen, indem man in beiden Fällen `number` in das Ausführungsfenster eingibt. Im ersten Fall erhält man die Antwort 5, im zweiten Fall eine Fehlermeldung.

Sichtbarkeit: Im ersten Fall kann man überall (nach dem `define`) auf den Wert, der `number` zugewiesen wurde, zugreifen. Im zweiten Fall ist `number` nur innerhalb der Let-Klausel sichtbar.