
Programmieren für Geisteswissenschaftler

Blatt 1

Abgabe: 8.11.2001

1. Welche Resultate druckt der Scheme-Interpreter als Antwort auf die Eingabe folgender Ausdrücke? Nimm dabei an, daß die Ausdrücke nacheinander in einer Sitzung eingegeben werden. (Ausprobieren liefert natürlich die richtigen Ergebnisse, hat aber geringeren geistigen Nährwert.)

23

(+ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20)

(+ (* 2 4) (- 4 6))

(define a 3)

(define b (+ a 1))

b

(= a b)

(define f

(lambda (x)

(+ x 1)))

(f b)

x

(define mike

(lambda (sperber)

(* sperber sperber)))

(define sperber

(lambda (mike)

(+ mike mike)))

(mike (sperber 5))

2. Sei C eine Temperatur in °C. Die entsprechende Temperatur F in Fahrenheit ergibt sich als:

$$F = \frac{9}{5}C + 32$$

Schreibe eine Prozedur namens `celsius->fahrenheit`, die als Parameter eine Temperatur in °C akzeptiert und eine Temperatur in Fahrenheit akzeptiert. Das Gerüst könnte so aussehen:

(define celsius->fahrenheit

(lambda (T)

...))

So sollte z.B. `(celsius->fahrenheit 5)` das Ergebnis 41 liefern.

Bonus: Schreibe eine Prozedur `fahrenheit->celsius`, welche das umgekehrte tut!

3. Schreibe eine Prozedur namens `min` mit zwei Parametern, welche als Ergebnis die kleinere zweier Zahlen liefert. Das Gerüst könnte so aussehen:

```
(define min
  (lambda (number-1 number-2)
    ...))
```

Bonus: Schreibe eine Prozedur namens `min-3` mit *drei* Parametern, die deren Minimum berechnet!

4. [5 Punkte] Schreibe eine Prozedur, die aus dem Preis für einen Liter Sprit und dem bezahlten Preis (in Pfennigen) einer Tankfüllung die zu berappende Öko-Steuer berechnet. Der Einfachheit halber sei die Ökosteuer mit 14 Pfennig pro Liter Sprit überschlagen. Binde die Prozedur an den Namen `oekosteuer`. In der REPL sollte sich die Prozedur beispielsweise so verhalten:

```
> (oekosteuer 203 10150)
700
```

Das Gerüst könnte so aussehen:

```
(define oekosteuer
  (lambda (preis-pro-liter bezahlter-betrag)
    ...))
```

5. Beschrifte in den folgenden Scheme-Ausdrücken die Teilausdrücke jeweils mit *syntaktisches Schlüsselwort*, *Parameter*, *Operator* und *Operand*. Der Ausdruck

```
((lambda (x y) (+ x y)) 1 2)
```

wäre also so zu annotieren:

$$\begin{array}{ccccccc}
 & & \text{Operator} & & & \text{Operanden} & \\
 \overbrace{((\underbrace{\text{lambda}}_{\text{synt. Schlüsselw.}} (\underbrace{x\ y}_{\text{Parameter}}) (\underbrace{+}_{\text{Operator}} (\underbrace{x\ y}_{\text{Operanden}}))) \overbrace{1\ 2}_{\text{Operanden}})} & & & & & & \\
 \text{synt. Schlüsselw.} & \text{Parameter} & \text{Operator} & \text{Operanden} & & &
 \end{array}$$

Du kannst die Bezeichnungen natürlich abkürzen.

- (a) `(+ 1 2)`
- (b) `(+ (* 3 4) 5)`
- (c) `((lambda (dont-touch-me) dont-touch-me) 4)`
- (d) `((lambda (winner loser) winner) -1 1)`
- (e) `(define add-one (lambda (number) (+ number 1)))`
- (f) `(add-one 3)`
- (g) `((lambda (apply-me-to-four)
 (apply-my-to-four 4))
 (lambda (double-me)
 (* double-me 2)))`