Mathematisches Institut
der Universität München
Prof. Dr. Helmut Schwichtenberg
Dr. Christian Urban

Wintersemester 2004/2005
Blatt 1

Übungen zum Schemekurs

Aufgabe 1. What does the Scheme interpreter return when evaluating the following expressions?

```
27

(+ 4 7 9)

(- 8 2)

(/ 8 4)

(* (+ 3 1) (- 6 2))

(define a 3)

(define b (+ a 1))

(+ a b (* a b))

(= a b)

(if (and (> a b) (< b (* a b)))
    b
    a)

(cond ((= a 4) 6)
      ((= b 4) (+ 6 7 a))
      (else 25))

(+ 2 (if (> b a) b a))
```

Lösung.
Just try them out.

Aufgabe 2. Write a Scheme-program that for three given natural numbers returns the sum of the squares of the two biggest given numbers.

Lösung.

```
; First Solution

(define (sum-of-squares x y)
  (+ (* x x) (* y y)))

(define (exercise2 a b c)
  (sum-of-squares (max a b c) (mean a b c)))

(define (mean a b c)
  (cond ((or (<= a b c) (<= c b a)) b)
        ((or (<= b a c) (<= c a b)) a)
        (else c)))

(exercise2 2 3 4)
(exercise2 5 3 4)

; notice:

(or #t #t #t #f)
(and #t #t #t #f)

(mean 7 3 5)
(mean 7 4 1)
(mean 2 9 5)

; Second Solution:

(define (exercise2 a b c)
  (if (< a b)
      (if (< a c)
          (sum-of-squares b c)
          (sum-of-squares a b))
      (if (< b c)
          (sum-of-squares a c)
          (sum-of-squares a b))))

(exercise2 2 3 4)
(exercise2 5 3 4)

; Third Solution

(define (exercise2 a b c)
  (if (< a b)
      (sum-of-squares b (if (< a c)
                            c
                            a))
      (sum-of-squares a (if (< b c)
                            c
                            b))))

(exercise2 2 3 4)
(exercise2 5 3 4)
```

Aufgabe 3. Describe the differences (if there are any) between the build-in `if` and the following program defining `new-if`:

```
(define (new-if test consequent alternative)
  (cond (test consequent)
        (else alternative)))
```

Lösung.
Try the following:

```
(if #t 27)
(new-if #t 27)

(if #t 5 (loop))
(new-if #t 5 (loop))

(if #f x 27)
(new-if #f x 27)

(define (loop) (loop))
(loop)
```

Aufgabe 4. Describe how the following program works and explain its output.

```
(define (apply-to f x) (f x))

(apply-to (lambda (x)
            (display "world !")
            (display x)
            (newline))
          (display "hello "))
```

You can use `(trace apply-to)` in order to see the execution trace.

Aufgabe 5. Give a recursive *and* an iterative program for the function defined as follows:

$$\begin{aligned} &f(0) = f(1) = f(2) = 1 \\ &f(n) = f(n-1) + 2f(n-2) + 3f(n-3) \quad (n \geq 3) \end{aligned}$$

Again, use `(trace f)` to see the execution trace.

Lösung.

```
; recursive

(define (f n)
  (cond ((= n 0) 1)
        ((= n 1) 1)
        ((= n 2) 1)
        (else
         (+ (f (- n 1)) (* 2 (f (- n 2))) (* 3 (f (- n 3)))))))

; or

(define (f n)
  (if (<= n 2)
      1
      (+ (f (- n 1)) (* 2 (f (- n 2))) (* 3 (f (- n 3))))))

(trace f)
(f 5)

; iterative

(define (f-iter n)
  (if (<= n 2)
      1
      (f-aux n 2 1 1 1)))


(define (f-aux n count acc1 acc2 acc3)
  (if (= count n)
      acc1
      (f-aux n (+ count 1) (+ acc1 (* 2 acc2) (* 3 acc3)) acc1 acc2)))

(trace f-aux)
(f-iter 5)
```

Abgabe. Hand in the exercises on Tuesday, October 5th, at the beginning of the lecture or via email to `urban@mathematik.uni-muenchen.de`.