

# Übungen zu Softwareentwicklung III, Funktionale Programmierung

Blatt 2, Woche 3

Leonie Dreschler-Fischer

WS 2017/2018

**Ausgabe:** Freitag, 27.10.2017,

**Abgabe der Lösungen:** bis Montag, 13.11.2017, 12:00 Uhr per email bei den Übungsgruppenleitern.

**Ziel: Namen, Symbolverarbeitung und exakte Zahlen:** Die Aufgaben auf diesem Zettel dienen dazu, sich mit der Definition von lokalen Variablen sowie den Gültigkeitsbereichen von definierten Namen vertraut zu machen. Außerdem üben Sie die Verwendung von *special form expressions*.

Weiterhin üben Sie, einfache Iterationen durch Rekursion auszudrücken, sowie das Rechnen mit beliebiger Genauigkeit mittels Rationalzahlen.

**Bearbeitungsdauer:** Die Bearbeitung sollte insgesamt nicht länger als 4 Stunden dauern.

**Homepage:**

[http://kogs-www.informatik.uni-hamburg.de/~dreschle/teaching/Uebungen\\_Se\\_III/Uebungen\\_Se\\_III.html](http://kogs-www.informatik.uni-hamburg.de/~dreschle/teaching/Uebungen_Se_III/Uebungen_Se_III.html)

Bitte denken Sie daran, auf den von Ihnen eingereichten Lösungsvorschlägen *Ihren Namen und die Matrikelnummer, den Namen der Übungsgruppenleiterin / des Übungsgruppenleiters und Wochentag und Uhrzeit der Übungsgruppe* anzugeben, damit wir ihre Ausarbeitungen eindeutig zuordnen können.

# 1 Symbole und Werte, Umgebungen

Bearbeitungszeit 1 Std., 9 Pkt.

Gegeben seien die folgenden Definitionen:

```
(define wuff 'Flocki)
(define Hund wuff)
(define Wolf 'wuff)

(define (welcherNameGiltWo PersonA PersonB)
  (let ((PersonA 'Zaphod)
        (PersonC PersonA))
    PersonC))

(define xs1 '(0 2 3 wuff Hund))
(define xs2 (list wuff Hund))
(define xs3 (cons Hund wuff))
```

Zu welchen Werten evaluieren dann die folgenden Ausdrücke? Begründen Sie (kurz) die Antworten.

1. wuff
2. Hund
3. Wolf
4. (**quote** Hund)
5. (**eval** Wolf)
6. (**eval** Hund)
7. (**eval** 'Wolf)
8. (welcherNameGiltWo 'lily 'potter)
9. (caddr xs1)
10. (cdr xs2)
11. (cdr xs3)
12. (**sqrt** 1/4)

13. (`eval` '(welcherNameGiltWo 'Wolf 'Hund))

14. (`eval` (welcherNameGiltWo 'Hund 'Wolf ))

## 2 Rechnen mit exakten Zahlen:

Bearbeitungszeit 2 1/2 Std.

### 2.1 Die Fakultät einer Zahl

2 Pkt.

Definieren sie eine rekursive Funktion zur beliebig genauen Berechnung der Fakultät  $n!$  einer natürlichen Zahl  $n$ .

$$\begin{aligned} 0! &= 1 \\ n! &= n \cdot (n-1)!, n \in \mathbb{N} \end{aligned}$$

### 2.2 Potenzen von Rationalzahlen

3 Pkt.

Definieren Sie eine rekursive Funktion (`power r n`), die für Rationalzahlen  $r \in \mathbb{Q}$  und ganzzahlige Exponenten  $n \in \mathbb{N}$  die Potenz  $r^n$  mit beliebiger Genauigkeit errechnet.

Verwenden Sie das folgende Rekursionsschema:

$$\begin{aligned} r^0 &= 1 \\ r^n &= \begin{cases} r^{n-1} \cdot r & , n \text{ ungerade} \\ (r^{\frac{n}{2}})^2 & , n \text{ gerade} \end{cases} \end{aligned}$$

**Hinweis:** In DrRacket sind die folgenden Funktionen vordefiniert:

(`even? n`)  $n$  gerade?

(`odd? n`)  $n$  ungerade?


(`sqr x`) Quadrat von  $x$

## 2.3 Die Eulerzahl $e$ :

6 Pnkt.

Berechnen Sie die Eulerzahl  $e$  mittels der folgenden Reihe auf 1000 Stellen genau (d.h. bis das letzte Glied der Reihe  $< \frac{1}{10^{1000}}$  ist).

$$2 \cdot e = 1 + \frac{2}{1!} + \frac{3}{2!} + \frac{4}{3!} + \frac{5}{4!} + \frac{6}{5!} + \dots$$

 **Anmerkung:** Anzeige des Ergebnisses: DrRacket wird Ihnen das Ergebnis als Quotient zweier teilerfremder Zahlen anzeigen. Um wirklich die ersten 1000 Ziffern zu sehen, multiplizieren Sie einfach das Ergebnis mit  $10^{1001}$ .

## 2.4 $\pi$ :

4 Zusatz-  
pnkt.

Wer noch Lust auf mehr exakte Arithmetik hat, kann sich auch noch an der folgenden Reihe versuchen:

Berechnen Sie die ersten Stellen der Zahl  $\pi$  nach der Formel von Gregory und Leibniz (auf soviele Stellen, wie Ihre Geduld und Ihr Speicher hergeben):

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} \dots$$

Warum konvergiert die Näherungsfolge für  $e$  schneller als die von  $\pi$ ?

## 3 Typprädikate:

30 Minuten, 5 Pnkt.

In Racket gibt es die Typprädikate

`boolean?`, `pair?`, `list?`, `symbol?`, `number?`,  
`char?`, `string?`, `vector?`, `procedure?`.

Verwenden Sie diese Prädikate, um eine polymorphe Funktion `type-of` zu definieren, die für einen gegebenen Ausdruck den Typ ermittelt. Berechnen Sie die Werte der folgenden Ausdrücke und erläutern Sie die Ergebnisse:

```
(type-of (* 2 3 4))  
(type-of (not 42))  
(type-of '(eins zwei drei))  
(type-of '())  
(define (id z) z)  
(type-of (id sin))  
(type-of (string-ref "SE3" 2))  
(type-of (lambda (x) x))
```

(type-of type-of)  
(type-of (type-of type-of))

**Erreichbare Punkte:** 25

**Erreichbare Zusatzunkte:** 4