Aufgaben – ausgewählte Lösungen

Einführung in die Programmierung

Johannes Brauer

29. Januar 2021

Inhaltsverzeichnis

1 Erste Schritte in Racket

Machen Sie sich mit den NORDAKADEMIE-Rechnern vertraut und richten Sie Ihren Arbeitsplatz ein (Mail, Webbrowser, Verzeichnisse für die Vorlesungen usw.).

Finden und starten Sie DrRacket nach den Anweisungen in der Vorlesung. Werten Sie einen ersten Ausdruck aus, z. B. (* 6 7).

Welche Funktion haben die Buttons? Welche Menü-Befehle verstehen Sie schon?

Schauen Sie sich in einem Webbrowser die Seiten zu Racket unter `https://racket-lang.org/` an. Wo finden Sie Hilfe zur Bedienung von DrRacket? Wie können Sie sich über die Sprache Racket informieren? Wo finden Sie alle vordefinierten mathematischen Funktionen?

2 Auswertung arithmetischer Ausdrücke

a. Wie wird der Ausdruck
`(* (+ 2 2) (/ (* (+ 3 5) (/ 30 10)) 2))`
ausgewertet?

b. Experimentieren Sie mit verschiedenen Operatoren und Zahlenarten.

c. Werten Sie die folgenden Ausdrücke aus und vergleichen Sie die Resultate:

```
(- 1.0 0.9)
(- 1000.0 999.9)
(- #i1000.0 #i999.9)
```

3 Aufschreiben elementarer Funktionen

Schreiben Sie für die folgenden mathematischen Formeln Racket-Funktionsdefinitionen auf:

a. $n^2 + 1$

```
(define fa
  (lambda [n]
    (+ (* n n) 1)))
```

b. $\frac{1}{2}n^2 + 3$

```
(define fb
  (lambda [n]
    (+ (/ (* n n) 2) 3)))
```

c. $2 - \frac{1}{n}$

```
(define fc
  (lambda [n]
    (- 2 (/ 1 n))))
```

Geben Sie die Racket-Funktionen in das Definitionsfenster von DrRacket ein. Geben Sie anschließend in das Interaktionsfenster Funktionsaufrufe für diese Funktionen ein.

4 Aufschreiben elementarer Funktionen

In der Praxis findet der Programmierer selten mathematische Formeln vor. Aufgabenstellungen sind eher als Prosatext gegeben. Die Berechnungsformeln muss er selbst entwickeln durch

- eigenes Nachdenken,
- Nachschlagen in geeigneten Quellen oder
- Nachfragen beim Auftraggeber.

Finden Sie für die folgenden Aufgabenstellungen die passenden Formeln und schreiben Sie diese als Funktionsdefinitionen in *Racket* auf:

a. Berechnung des Rauminhalts eines Quaders aus dessen Länge, Breite und Höhe.

```
(define quader-volumen
  (lambda [laenge breite hoehe]
    (* laenge breite hoehe)))
```

b. Schreiben Sie eine Funktion, die aus der Entfernung und den Geschwindigkeiten zweier Züge die Zeit ermittelt, nach der die Züge sich treffen, wenn Sie sich auf einem gemeinsamen Streckenabschnitt von ihren jeweiligen Startpunkten aus aufeinander zu bewegen.

```
(define zug-treffen
  (lambda [entfernung geschwndgkt1 geschwndgkt2]
    (/ entfernung (+ geschwndgkt1 geschwndgkt2))))
```

c. Berechnung der Miete, die eine Spielerin in Monopoly bezahlen muss, falls sie auf einen Bahnhof trifft, der einer anderen Spielerin gehört. Die Miete ist davon abhängig wie viele Bahnhöfe der anderen Spielerin gehören:

Anzahl der Bahnhöfe	Miete
1	500
2	1000
3	2000
4	4000

Hinweis: Ein Aufruf `(expt x y)` liefert x^y als Ergebnis.

```
(define bahnhofsmiete
  (lambda [anzahl-bahnhoefe]
    (* 500 (expt 2 (- anzahl-bahnhoefe 1)))))
```

5 Anwenden der Aufschreibregeln

Schreiben Sie die Funktion zur Berechnung der Bahnhofsmiete in Monopoly (s. o.) gemäß den Regeln 1 bis 3 aus der Vorlesung auf.

```
;; berechnet die Bahnhofsmiete aus der Anzahl der Bahnhöfe
(define bahnhofsmiete
  (lambda [anzahl-bahnhoefe]
    (* 500 (expt 2 (- anzahl-bahnhoefe 1)))))
;; Beispielanwendungen
(= (bahnhofsmiete 4) 4000)
(= (bahnhofsmiete 3) 2000)
(= (bahnhofsmiete 2) 1000)
(= (bahnhofsmiete 1)  500)
```

Wenn nichts anderes angeben ist, sind auch die Funktionen für die folgenden Aufgaben gemäß diesen Regeln aufzuschreiben!!!

6 Profit für den Kinobesitzer

Ein altmodisches Vorstadtkino besitzt eine einfache Formel für die Berechnung des Profits einer Vorstellung: Jeder Kinobesucher bezahlt 500 Währungseinheiten für die Eintrittskarte. Jede Vorstellung kostet das Kino 2000 Währungseinheiten plus 50 Währungseinheiten pro Besucher. Schreiben Sie eine Funktion `profit` zur Berechnung des Profits bei gegebener Besucherzahl.

```
;; berechnet den Profit eine Filmvorfuehrung aus 2000 WE Fixkosten
;; und 50 WE Kosten pro Besucher bei gegebener Besucherzahl und
;; 500 WE Kartenpreis
(define profit
  (lambda [besucherzahl]
    (- (* 500 besucherzahl)
       (+ (* 50 besucherzahl) 2000))))
;; Beispielanwendungen
(= (profit 100) 43000)
```

7 Modifikation von `kino.rkt`

a. Modifizieren Sie das Programm `kino.rkt` so, dass die Fixkosten einer Veranstaltung wegfallen und dafür 15 Währungseinheiten pro Besucher an Kosten anfallen.

b. Nehmen Sie die gleiche Modifikation auch an der Funktion `profit` aus der Vorlesung vor, die ohne Hilfsfunktionen auskommt, und vergleichen Sie die Ergebnisse.

8 Einsatz von Hilfsfunktionen

Die folgenden Aufgaben sind unter Benutzung von Hilfsfunktionen zu lösen. Befolgen Sie unbedingt alle in der Vorlesung angegebenen Regeln:

a. Schreiben Sie ein Programm, das das Volumen eines Zylinders zu berechnen erlaubt. Eingangsgrößen sind der Radius und die Höhe des Zylinders.

b. Schreiben Sie ein Programm, das die Oberfläche eines Zylinders zu berechnen erlaubt. Eingangsgrößen sind der Radius und die Höhe des Zylinders.

c. Schreiben Sie ein Programm, das die Oberfläche eines Rohrs zu berechnen erlaubt. Eingangsgrößen sind der Innenradius, die Wandstärke und die Länge des Rohrs.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;; Lösung von Simon Greßmann, I18b ;;;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;Aufgabe 8a
(define mein-pi 3.14)
;Berechnet Kreisfläche aus Radius
(define Kreisflaeche
  (lambda [radius]
    (* radius radius mein-pi)))
(= (Kreisflaeche 1) 3.14)
(= (Kreisflaeche 2) 12.56)
;Berechnet das Volumen eines Zylinders in Abhängigkeit von Radius und Höhe
(define Zylindervolumen
  (lambda [willi hoehe]
    (* (Kreisflaeche willi) hoehe)))
(= (Zylindervolumen 1 1) 3.14)

;Aufgabe 8b
;Berechnet den Umfang eines Kreises in Anhängigkeit von Pi.
(define Kreisumfang
  (lambda [radius]
    (* 2 mein-pi radius)))
(= (Kreisumfang 1) 6.28)

;Berechnet die Mantelfläche eines Zylinders in Abhängigkeit von Radius
;und Höhe
(define Mantelflaeche
  (lambda (radius hoehe)
    (* (Kreisumfang radius) hoehe)))
(= (Mantelflaeche 1 1) 6.28)
(= (Mantelflaeche 2 1) 12.56)

;Berechnet die Oberfläche eines Zylinders in Abhängigkeit von Radius und Höhe
(define Zylinderoberflaeche
  (lambda [radius hoehe]
    (+ (* 2 (Kreisflaeche radius)) (Mantelflaeche radius hoehe))))
(= (Zylinderoberflaeche 1 1) 12.56)

;Aufgabe 8c
;Berechnet die Fläche eines Kreisrings in Abhängigkeit von innenradius und
;Breite
(define Kreisringflaeche
  (lambda [innenradius breite]
    (- (Kreisflaeche (+ innenradius breite)) (Kreisflaeche innenradius))))
(= (Kreisringflaeche 1 1) 9.42)
;Berechnet die Oberfläche eines Rohrs in Abhängigkeit von dessen Inneradius,
;Wandstärke und Länge.
(define Rohroberflaeche
  (lambda [innenradius wandstaerke laenge]
    (+ (Mantelflaeche (+ innenradius wandstaerke) laenge)
```

```
      (Mantelflaeche innenradius laenge)
      (* 2 (Kreisringflaeche innenradius <dstaerke)))))
(= (Rohroberflaeche 1 1 1) 37.68)
```

9 Ersetzungsmodell

Gegeben sei die folgende Funktionsdefinition:

```
(define f
   (lambda [x y]
      (+ (* 3 x) (* y y))))
```

Werten Sie die folgenden Ausdrücke Schritt für Schritt unter Anwendung des Ersetzungsmodells aus:

a. `(f 1 (* 2 3))`

```
(f 1 (* 2 3))
= ((lambda [x y] (+ (* 3 x) (* y y))) 1 (* 2 3))
= ((lambda [x y] (+ (* 3 x) (* y y))) 1 6)
= (+ (* 3 1) (* 6 6))
= (+ 3 36)
= 39
```

b. `(+ (f 1 2) (f 2 1))`

```
(+ (f 1 2) (f 2 1))
= (+ ((lambda [x y] (+ (* 3 x) (* y y))) 1 2)
     ((lambda [x y] (+ (* 3 x) (* y y))) 2 1))
= (+ (+ (* 3 1) (* 2 2))
     (+ (* 3 2) (* 1 1)))
= (+ (+ 3 4) (+ 6 1))
= (+ 7 7)
= 14
```

Ausführlich:

```
(+ (f 1 2) (f 2 1))
= (+ ((lambda [x y] (+ (* 3 x) (* y y))) 1 2)
     (f 2 1))
= (+ ((lambda [x y] (+ (* 3 x) (* y y))) 1 2)
     ((lambda [x y] (+ (* 3 x) (* y y))) 2 1))
= (+ (+ (* 3 1) (* 2 2))
     (+ ((lambda [x y] (+ (* 3 x) (* y y))) 2 1))
= (+ (+ (* 3 1) (* 2 2))
     (+ (* 3 2) (* 1 1)))
= (+ (* 3 4) (+ (* 3 2) (* 1 1)))
= (+ (+ 3 4) (+ 6 1))
= (+ 7 (+ 6 1))
= (+ 7 7)
= 14
```

c. `(f (f 1 (* 2 3)) 19)`

```
(f (f 1 (* 2 3)) 19)
= ((lambda [x y] (+ (* 3 x) (* y y))) (f 1 (* 2 3)) 19)
= ((lambda [x y] (+ (* 3 x) (* y y)))
     ((lambda [x y] (+ (* 3 x) (* y y))) 1 (* 2 3)) 19)
= ((lambda [x y] (+ (* 3 x) (* y y)))
     ((lambda [x y] (+ (* 3 x) (* y y))) 1 6) 19)
= ((lambda [x y] (+ (* 3 x) (* y y)))
     (+ (* 3 1) (* 6 6)) 19)
= ((lambda [x y] (+ (* 3 x) (* y y))) (+ 3 36) 19)
```

```
= ((lambda [x y] (+ (* 3 x) (* y y))) 39 19)
= (+ (* 3 39) (* 19 19))
= (+ 117 361)
= 478
```

Als Alternative hier noch eine Schreibarbeit sparende Lösung von Alessandra Blank:

```
; Nebenrechnung für x:
  sei x = (f 1 (* 2 3))
  x = (f 1 6)
    = ((lambda [x y] (+ (* 3 x) (* y y))) 1 6)
    = (+ (* 3 1) (* 6 6)
    = (+ 3 36)
   x = 39

  (f 39 19)
  = ((lambda [x y] (+ (* 3 x) (* y y))) 39 19)
  = (+ (* 3 39) (* 19 19))
  = (+ 117 361)
  = 478
```

10 Bedingte Funktion

Schreiben Sie ein Programm, das aus dem Bruttoeinkommen eines Arbeitnehmers, das sich aus der Anzahl der Arbeitsstunden und seinem Bruttostundenlohn ergibt, sein Nettoeinkommen durch Abzug der Einkommensteuer berechnet. Die Einkommensteuer wird dabei nach einem steuererklärungaufbierdeckelgeeigneten Tarif ermittelt, der folgendermaßen definiert ist:

Einkommen	**Steuersatz** [%]
$<= 5000$	0
> 5000 *und* ≤ 10000	15
> 10000 *und* ≤ 100000	29
> 100000	64

Der Steuersatz gilt immer nur für die Einkommensanteile in dem jeweiligen Intervall.
Die Funktion `nettoeinkommen` soll nach folgendem Schema aufrufbar sein:

```
(nettoeinkommen anzahl-arbeitsStunden stundenLohn)
```

Hier noch ein paar Testvorgaben:

```
;; Beispielanwendungen
(= (nettoeinkommen 1 5001) 5000.85)
(= (nettoeinkommen 1 10001) 9250.71)
(= (nettoeinkommen 1 100001) 73150.36)
```

Hinweise:

a. Lesen Sie den Aufgabentext aufmerksam durch. Jeder Satz bedeutet etwas.

b. Entwickeln Sie die Funktion gemäß den Regel 1 bis 6. Benutzen Sie Hilfsfunktionen und machen von Variablendefinitionen (benannte Konstanten, Regel 5) Gebrauch.

```
;; Steuertabelle gemäß Aufgabenstellung
(define steuergrenzeI 5000)
(define steuergrenzeII 10000)
(define steuergrenzeIII 100000)

(define steuersatz1 0)
(define steuersatz2 15/100)
(define steuersatz3 29/100)
```

```
(define steuersatz4 64/100)

;; feste Steuerbeträge gemäß obiger Tabelle
(define steuern-fuer-steuergrenzeI (* steuersatz1 steuergrenzeI))
(define steuern-fuer-steuergrenzeII
  (+ steuern-fuer-steuergrenzeI
     (* steuersatz2 (- steuergrenzeII steuergrenzeI))))
(define steuern-fuer-steuergrenzeIII
  (+ steuern-fuer-steuergrenzeII
     (* steuersatz3 (- steuergrenzeIII steuergrenzeII))))

;; Ermittlung des Einkommenssteuersatzes aus dem Einkommen gemäß obiger Tabelle
(define steuersatz
  (lambda [einkommen]
    (cond
      [(and (<= einkommen steuergrenzeI) (>= einkommen 0)) steuersatz1]
      [(and (> einkommen steuergrenzeI) (<= einkommen steuergrenzeII)) steuersatz2]
      [(and (> einkommen steuergrenzeII) (<= einkommen steuergrenzeIII)) steuersatz3]
      [(> einkommen steuergrenzeIII) steuersatz4])))
;; Beispielanwendungen
"steuersatz"
(= (steuersatz 2000) 0)
(= (steuersatz steuergrenzeI) 0)
(= (steuersatz 7500) 15/100)
(= (steuersatz steuergrenzeII) 15/100)
(= (steuersatz 50000) 29/100)
(= (steuersatz steuergrenzeIII) 29/100)
(= (steuersatz 1000000) 64/100)

;; Berechnung des Bruttoeinkommens eines Arbeitnehmers
;; der Anzahl der Arbeitsstunden und dem Bruttostundenlohn
(define bruttoeinkommen
  (lambda [arbeitsstunden stundenlohn]
    (* arbeitsstunden stundenlohn)))
;; Beispielanwendungen
"bruttoeinkommen"
(= (bruttoeinkommen 10 5) 50)

;; Ermittlung der Einkommensteuer aus dem einkommen
;; gemäß Steuertarif in Aufgabenstellung
(define einkommensteuer
  (lambda [einkommen]
    (cond
      [(= (steuersatz einkommen) steuersatz1)
       (* steuersatz1 einkommen)]
      [(= (steuersatz einkommen) steuersatz2)
       (+ steuern-fuer-steuergrenzeI
          (* steuersatz2 (- einkommen steuergrenzeI)))]
      [(= (steuersatz einkommen) steuersatz3)
       (+ steuern-fuer-steuergrenzeII
          (* steuersatz3 (- einkommen steuergrenzeII)))]
      [(= (steuersatz einkommen) steuersatz4)
       (+ steuern-fuer-steuergrenzeIII
          (* steuersatz4 (- einkommen steuergrenzeIII)))])))
;; Beispielanwendungen
"einkommensteuer"
(= (einkommensteuer 2000) 0)
(= (einkommensteuer steuergrenzeI) 0)
(= (einkommensteuer 8000) 450)
```

```
(= (einkommensteuer steuergrenzeII) 750)
(= (einkommensteuer 50000) 12350)
(= (einkommensteuer steuergrenzeIII) 26850)
(= (einkommensteuer 100100) (+ 26850 64))

;; Berechnung des Nettolohns eines Arbeitnehmers aus
;; Anzahl Arbeisstunden und Bruttostundenlohn (in Euro)
(define nettoeinkommen
  (lambda [arbeitsstunden stundenlohn]
    (- (bruttoeinkommen arbeitsstunden stundenlohn)
       (einkommensteuer (bruttoeinkommen arbeitsstunden stundenlohn)))))
;; Beispielanwendungen
"nettoeinkommen"
(= (nettoeinkommen 1 5001) (/ 500085 100))
(= (nettoeinkommen 1 10001) (/ 925071 100))
(= (nettoeinkommen 1 100001) (/ 7315036 100))
```

11 Zusatzaufgaben zu bedingten Funktionen

a. Eine Kreditkartengesellschaft gewährt ihren Kunden nach Jahresumsatz gestaffelte Rückerstattung von Kreditkartenbelastungen. Die Rückerstattungen könnten z. B. wie folgt aussehen:

- ein viertel Prozent für die ersten 500€ des Jahresumsatzes (nur Belastungen keine Gutschriften werden gezählt),
- ein halbes Prozent für die nächsten 1000€, d. h. für den Umsatzanteil zwischen 500€ und 1500€,
- ein dreiviertel Prozent für die nächsten 1000€, d. h. für den Umsatzanteil zwischen 1500€ und 2500€ und
- ein Prozent für die Umsatzanteile oberhalb von 2500€.

Ein Kunde mit einem Umsatz von 400€ erhält demnach eine Gutschrift von 1€ ($= \frac{1}{4} \cdot \frac{1}{100} \cdot 400$). Ein Kunde mit einem Umsatz von 1400€ erhält eine Gutschrift von 5,75€:

- 1,25€ ($= \frac{1}{4} \cdot \frac{1}{100} \cdot 500$) für die ersten 500€ plus
- 4,50€ ($= \frac{1}{2} \cdot \frac{1}{100} \cdot 900$) für die nächsten 900€

Lösen Sie die folgenden Teilaufgaben

(a) Bestimmen Sie manuell die Gutschriften für Umsätze 2000€ und 2600€.

(b) Schreiben Sie eine Funktion `rueckerstattung`, die einen Umsatz als Argument akzeptiert und den Rückerstattungsbetrag ermittelt.

```
;; berechnet die Rückerstattung für Umsätze zwischen 0 and 500
(define rueckerstattung-0-500
  (lambda [a]
    (* a (* .25 1/100))))
;; Beispielanwendungen
(= (rueckerstattung-0-500 400) 1)

;; berechnet die Rückerstattung für Umsätze zwischen 500 and 1500
(define rueckerstattung-500-1500
  (lambda [a]
    (+ (rueckerstattung-0-500 500)
       (* (- a 500) (* .50 1/100)))))
;; Beispielanwendungen
(= (rueckerstattung-500-1500 1400) 5.75)

;; berechnet die Rückerstattung für Umsätze zwischen 1500 and 2500
(define rueckerstattung-1500-2500
  (lambda [a]
```

```
      (+ (rueckerstattung-500-1500 1500)
         (* (- a 1500) (* .75 1/100)))))
;; Beispielanwendungen
(= (rueckerstattung-1500-2500 2000) 10.00)

;; berechnet die Rückerstattung für Umsätze von 2500 und höher
(define rueckerstattung-2500+
  (lambda [a]
    (+ (rueckerstattung-1500-2500 2500)
       (* (- a 2500) (* 1 1/100)))))
;; Beispielanwendungen
(= (rueckerstattung-2500+ 2600) 14.75)

;; berechnet Rückerstattungsbetrag für Kreditkarteninhaber
;; bei einem bestimmten Jahresumsatz
(define rueckerstattung
  (lambda [umsatz]
    (cond
      [(<= umsatz 500)
       (rueckerstattung-0-500 umsatz)]
      [(and (> umsatz 500) (<= umsatz 1500))
       (rueckerstattung-500-1500 umsatz)]
      [(and (> umsatz 1500) (<= umsatz 2500))
       (rueckerstattung-1500-2500 umsatz)]
      [else
       (rueckerstattung-2500+ umsatz)])))
;; Beispielanwendungen
(= (rueckerstattung 400) 1)
(= (rueckerstattung 1400) 5.75)
(= (rueckerstattung 2000) 10.00)
(= (rueckerstattung 2600) 14.75)
```

b. Wieviele reelle Lösungen besitzt eine quadratische Gleichung

$$ax^2 + bx + c = 0$$

für beliebige Koeffizienten a, b und c?

(a) Betrachten Sie zunächst nur *echte* quadratische Gleichungen, d. h. es gilt $a \neq 0$

(b) Erweitern Sie die Lösung so, dass auch der Fall $a = 0$ korrekt behandelt wird.

Lösung: Die Lösungen der quadratischen Gleichung

$$ax^2 + bx + c = 0$$

können für den Fall $a \neq 0$ mit der Formel

$$L_q : x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

berechnet werden. Für den Fall $a = 0$ gibt es eine Lösung $-c/b$, vorausgesetzt $b \neq 0$.

```
;; berechnet der Radikanden der Lösungsformel
(define radikand
  (lambda [a b c]
    (- (sqr b) (* 4 a c))))
;: Beispielanwendungen
(= (radikand 1 2 1) 0)
(= (radikand 1 1 1) -3)
(= (radikand 1 3 1) 5)
```

```
;; berechnet die Anzahl der Lösungen einer quadratischen Gleichung mit
;; den Koeffizienten a, b und c
(define anzahl-loesungen
  (lambda [a b c]
    (cond
      [(= 0 a) (cond
                 [(= 0 b c) "unendlich viele Lösungen"]
                 [(and (= 0 b) (not (= 0 c))) "keine Lösung"]
                 [else "eine Lösung"])]
      [else (cond [(> (radikand a b c) 0) "zwei Lösungen"]
                  [(= (radikand a b c) 0) "eine Lösung"]
                  [else "keine Lösung"])])))
;; Beispielanwendugen
(string=? (anzahl-loesungen 0 0 0) "unendlich viele Lösungen")
(string=? (anzahl-loesungen 0 0 1) "keine Lösung")
(string=? (anzahl-loesungen 0 1 2) "eine Lösung")
(string=? (anzahl-loesungen 1 4 1) "zwei Lösungen")
(string=? (anzahl-loesungen 1 2 1) "eine Lösung")
(string=? (anzahl-loesungen 4 1 1) "keine Lösung")

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Funktion zur Berechnung der Lösungen einer quadratischen Gleichung;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;; Eine quadgl-loesung ist ein Wert
;;  (make-quadgl-loesung art x1 x2)
;; wobei art die Anzahl der Lösungen durch eine Zeichenkette
;; ("trivial" "keine" "eine" "zwei") repräsentiert wird und
;; x1, x2 ggf. die Lösungen enthalten
(define-struct quadgl-loesung [art x1 x2])

;; berechnet die Lösungen einer quadratischen Gleichung mit
;; den Koeffizienten a, b und c
(define loesungen
  (lambda [a b c]
    (cond
      [(= 0 a) (cond
                 [(= 0 b c)
                    (make-quadgl-loesung "trivial" 0 0)]
                 [(and (= 0 b) (not (= 0 c)))
                    (make-quadgl-loesung "keine" 0 0)]
                 [else (make-quadgl-loesung "eine" (/ (- c) b) 0)])]
      [else (cond [(> (radikand a b c) 0)
                   (make-quadgl-loesung
                       "zwei"
                       (/ (+ (- b) (sqrt (radikand a b c)))
                          (* 2 a))
                       (/ (- (- b) (sqrt (radikand a b c)))
                          (* 2 a)))]
                  [(= (radikand a b c) 0)
                   (make-quadgl-loesung "eine" (/ (- b) (* 2 a)) 0)]
                  [else (make-quadgl-loesung "keine" 0 0)])])))
;; Beispielanwendugen
(string=? (quadgl-loesung-art (loesungen 0 0 0)) "trivial")
(string=? (quadgl-loesung-art (loesungen 0 0 1)) "keine")
(and (string=? (quadgl-loesung-art (loesungen 0 1 2)) "eine")
     (= (quadgl-loesung-x1 (loesungen 0 1 2)) -2))
(and (string=? (quadgl-loesung-art (loesungen 1 4 3)) "zwei")
     (= (quadgl-loesung-x1 (loesungen 1 4 3)) -1)
```

```
        (= (quadgl-loesung-x2 (loesungen 1 4 3)) -3))
   (and (string=? (quadgl-loesung-art (loesungen 1 2 1)) "eine")
        (= (quadgl-loesung-x1 (loesungen 1 2 1)) -1))
   (string=? (quadgl-loesung-art (loesungen 4 1 1)) "keine")
```

12 Datenabstraktion

Befolgen Sie für die Lösung der Aufgabe die Regeln 7 und 8!

a. Definieren Sie eine Datenstruktur für „Zeitpunkte seit Mitternacht", die aus den Komponenten `stunden`, `minuten` und `sekunden` besteht.

   Entwickeln Sie eine Funktion `zeit->sekunden`, die eine Zeitpunkt-seit-Mittnacht-Struktur verarbeitet und die seit Mitternacht vergangenen Sekunden berechnet.

b. Definieren Sie geeignete **Datenstrukturen** für Kreise, die durch

   - die Koordinaten des Mittelpunkts und
   - den Radius

   gegeben sind.

   Schreiben Sie eine Funktion, die prüft, ob ein Punkt innerhalb eines Kreises liegt.

```
; Autor der Lösungen zu Aufgabe 12: David Lüder (I16b)
; Aufgabe 12a

; Eine Tageszeit ist ein Wert
; (make-daytime h m s)
; wobei h die Stunden, m die Minuten und s die Sekunden angeben,
; um die Uhrzeit zu definieren.
(define-struct daytime [h m s])

; gibt die absolute Zahl der vergangenen Sekunden seit Mitternacht
; bei gegebeher Tageszeit zurück.
(check-expect (zeit->sekunden (make-daytime 0 0 0)) 0)
(check-expect (zeit->sekunden (make-daytime 0 2 36)) 156)
(check-expect (zeit->sekunden (make-daytime 2 1 50)) 7310)
(check-expect (zeit->sekunden (make-daytime 23 59 59)) 86399)
(define zeit->sekunden
  (lambda [uhrzeit]
    (+ (* (daytime-h uhrzeit) 3600)
       (* (daytime-m uhrzeit) 60)
       (daytime-s uhrzeit))))

; Aufgabe 12b

; Ein Punkt ist ein Wert
; (make-point x y)
; wobei x und y Zahlen sind, welche die Position im Zweidimensionalen
; angeben.
(define-struct point [x y])

; Ein Kreis ist ein Wert
; (make-circle center radius)
; wobei center ein Punkt ist, welcher den Mittelpunkt darstellt und radius
; eine Zahl ist, welche den Radius des Kreises angibt.
(define-struct circle [center radius])

; berechnet den Abstand zwischen zwei übergebenen Punkten im Zweidimensionalen
(check-expect (point-distance (make-point 1 1) (make-point 1 2)) 1)
```

```
(check-expect (point-distance (make-point 7 2) (make-point 2 2)) 5)
(check-within (point-distance (make-point 5 5) (make-point 2 4)) 3.16 3.17)
(define point-distance
  (lambda [p1 p2]
    (sqrt (+ (sqr (- (point-x p1) (point-x p2)))
             (sqr (- (point-y p1) (point-y p2)))))))

; prüft, ob ein Punkt innerhalb eines Kreises liegt, dabei werden der Kreis
; und der Punkt übergeben
(check-expect (in-circle? (make-circle (make-point 10 10) 2)
                          (make-point 10 9)) #true)
(check-expect (in-circle? (make-circle (make-point 7 3) 5)
                          (make-point 2 2)) #false)
(check-expect (in-circle? (make-circle (make-point 9 2) 7)
                          (make-point 8 4)) #true)
(define in-circle?
  (lambda [circle point]
    (<= (point-distance (circle-center circle) point) (circle-radius circle))))
```