EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Praktische Informatik 1
Wintersemester 2026/27
Prof. Torsten Grust, Björn Bamberg, Nico Faden, Ludwig Kolesch
WSI — Database Systems Research Group

Übungsblatt 1

Abgabe bis spätestens **Mittwoch, den 28. Oktober 2026, 23:59 Uhr** auf Hand-In.

Wähle die richtige Sprachebene!
Bearbeitet die Aufgaben auf diesem Blatt mit der Sprachebene **Schreibe Dein Programm! — Anfänger**.

Stehst du auf dem Schlauch?
Steckst du irgendwo fest? Sind einige der Aufgaben unklar? Brauchst du den einen kleinen Schubs, um ins Rollen zu kommen? Schau in unser Forum—vielleicht findest du genau die Hilfe, die du brauchst.

Klausur-ähnliche Aufgaben
Die mit Ⓚ markierten Aufgaben sind ähnlich im Stil zu den Aufgaben, die ihr in der Klausur finden werdet. Nutzt diese Aufgaben also dazu, deine Erwartungen an die Klausur zu überprüfen und deine Fähigkeiten besser einzuschätzen.

Wie sieht eine Funktionsdefinition aus?
In dieser Vorlesung definieren wir Funktionen immer nach einem festen Schema. Dieses Schema besteht aus den fünf folgenden Teilen:

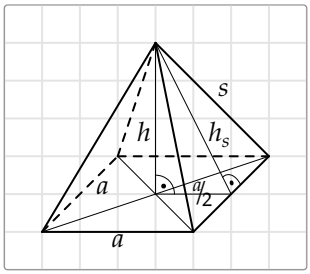
(1) einer Kurzbeschreibung (ein- bis zweizeiliger Kommentar) mit Bezug auf die Parameter(namen) und den Ergebniswert der Funktion,
(2) einer Signatur der Funktion,
(3) einigen Testfälle für die Funktion und
(4) einer Definition der Funktion mit ihrem
(5) Funktionsrumpf.

```
1 ; ...                                  ; (1)
2 (: f (... -> ...))                     ; (2)
3 (check-expect (f ...) ...)             ; (3)
4   ⋮
5 (check-within (f ...) ...)
6 (define f                              ; (4)
7   (lambda (p₁ p₂ ...)
8     ...))                              ; (5)
```

Aufgabe 1: Zusammenspiel von Funktionen Ⓚ

KOMPOSITION

Fast genauso lange wie die Menschen Pyramiden bauen (also schon ziemlich lange) suchen sie auch nach Möglichkeiten, ihre Pyramide als größte zu bezeichnen. Sei es das Volumen, die Grundfläche, die Höhe *etc.* — es gibt viele Maße, die dazu verwendet werden können. Eine der wilderen Optionen ist die Summe aller Kantenlängen der Pyramide — also die Summe aller Seiten der quadratischen Grundfläche a und den vier nach oben gehenden Kanten s. Ziel dieser Aufgabe ist, eine Funktion zu schreiben, die aus der Seitenlänge der Grundfläche a und der Höhe h genau diese Gesamtkantenlänge berechnet. Dazu gehen wir in mehreren Schritten vor.



Funktionskomposition
Große Funktionen sind schwer zu schreiben, zu lesen und zu verstehen. Daher ist es meist sinnvoll, mehrere kleine Funktionen zu schreiben und zusammenzusetzen, anstatt eine große Funktion zu schreiben. In dieser Aufgabe werdet ihr mehrere kleine Funktionen schreiben, die ihr, wenn möglich, in darauffolgenden Funktionen verwenden sollt.

A Schreibt eine Funktion `(: square-circumference (real -> real))`, die für eine gegebene Seitenlänge a den Umfang der quadratischen Grundfläche berechnet.

B Schreibt eine Funktion `(: pythagoras (real real -> real))`, welche den *Satz des Pythagoras* implementiert. Also $c = \sqrt{a^2 + b^2}$, wobei a, b die Arugmente der Funktion sind und c das Ergebnis ist.

Nützliches in Racket
Ihr könnt in Racket mittels der Funktion `(: sqrt (real -> real))` Wurzeln ziehen.

C Schreibt eine Funktion `(: pyramid-one-edge-length (real real -> real))`, die für eine gegebene Seitenlänge a und Höhe h die Länge einer Kante s berechnet (dazu müsst ihr die Länge h_s aus a und h berechnen).

D Schreibt schließlich eine Funktion `(: pyramid-edge-length (real real -> real))`, die für eine gegebene Seitenlänge a und Höhe h die Kantenlänge der Pyramide berechnet.

Aufgabe 2: Konversion von Einheiten Ⓚ

INVERSE FUNKTIONEN

Vielerorts auf der Welt werden die gleichen Dinge in unterschiedlichen Einheiten gemessen. Da die jeweiligen Einheiten aber grundlegend das Gleiche messen, kann man die Messungen von einer Einheit in die andere umrechnen. Ein klassisches Beispiel dafür sind Temperaturen. In den USA und wenigen weiteren Nationen wird die Temperatur in Grad Fahrenheit (°F) gemessen, während in Deutschland und vielen anderen Ländern die Temperatur in Grad Celsius (°C) gemessen wird. Eine Temperatur c in Grad Celsius lässt sich in eine Temperatur f in Grad Fahrenheit dabei wie folgt umrechnen:

$$f = \frac{9}{5}c + 32$$ 🌐.

A Schreibt eine Funktion `(: celsius->fahrenheit (real -> real))`, welche eine Temperatur in Grad Celsius in eine Temperatur in Grad Fahrenheit umrechnet.

B Schreibt eine Funktion `(: fahrenheit->celsius (real -> real))`, welche eine Temperatur in Grad Fahrenheit in eine Temperatur in Grad Celsius umrechnet.

C Schreibt einige Tests, die überprüfen, ob die Funktionen aus Teilaufgabe A und Teilaufgabe B *invers* zueinander sind.

> 🔄 Inverse Funktionen
> In der Mathematik gelten zwei Funktion f und g als *invers* zueinander wenn $f\big(g(x)\big) = x$ und $g\big(f(y)\big) = y$ für alle x und y gilt.

Aufgabe 3: Sauberer Code Ⓚ

CODEQUALITÄT

Vor dir liegt die Funktion `transform`, welche dein Kollege A. U. Weia nutzt, um eine gegebene natürliche Zahl `x` umzuwandeln:

```
(: transform (natural -> integer))
(define transform
  (lambda (x)
    (string->number (string-append "-" (number->string x) "E3"))))
```

> 🔥 Was machen die Funktionen?
> Die Funktion `transform` macht Verwendung von drei — in Racket eingebauten — Funktionen:
>
> - `string->number` 🌐 und `number->string` 🌐, welche Zeichenketten in Zahlen und wieder zurück umwandeln.
>
> ```
> (string->number "42")
> ; ↠ 42
> (number->string 42)
> ; ↠ "42"
> ```
>
> - `string-append` 🌐, welche mehrere Zeichenketten zu einer zusammenklebt.
>
> ```
> (string-append "Darth" " " "Va" "der")
> ; ↠ "Darth Vader"
> ```

Euer Kollege A. U. Weia hat weder Kommentare noch Testfälle für die Funktion `transform` geschrieben. Jetzt liegt es an euch, herauszufinden, was die Funktion macht und dies sinnvoll zu dokumentieren.

A Ruft `transform` mit verschiedenen Argumenten auf und versucht herauszufinden, was die Funktion macht.

B Schreibt dann einen Kommentar, der das Verhalten von `transform` beschreibt, und fügt einige Tests hinzu, die das Verhalten sinnvoll dokumentieren.

Beim Testen und Verstehen von `transform` sollte euch aufgefallen sein, dass die Funktion einige Defizite hat. Sie macht unnötige Umwandlungen von Zahlen in Zeichenketten und wieder zurück. Das ist nicht nur ineffizient, sondern auch fehleranfällig.

C Schreibt eine Funktion `better-transform`, die den Job von `transform` erledigt, aber den Umweg über Zeichenketten vermeidet. Vergesst Kommentare und Testfälle für eure Funktion nicht! 😉

> 💡 Andere Signatur?
> Sollte euer `better-transform` für mehr Argumente als `transform` funktionieren, passt die Signatur von `better-transform` entsprechend an.