Lehrstuhl für Kognitive Modellierung
Prof. Dr. Martin V. Butz
Dr. Jan Kneissler

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Informatik II

Programmierübungsblatt 0

Ausgabe: Do, 10.04.2014 Abgabe: 16te Kalenderwoche

Für die Bearbeitung der Übungsaufgaben haben Sie 90 Minuten Zeit. Bitte geben Sie Ihre Lösungen der Aufgaben rechtzeitig im Vorlesungssystem unter dem Punkt Präsenzübungen ab. Dazu müssen Sie sich mit Ihrem Benutzernamen und Passwort am Vorlesungssystem anmelden. Sie können mehrmals innerhalb der Bearbeitungszeit Ihre Lösungen abgeben, nur die letzte Version Ihrer Abgabe wird gespeichert. Wenn Sie mit den Übungen fertig sind, bleiben Sie bitte an ihrem Platz sitzen und warten, bis ein Tutor ihre Lösungen bewertet. **Bei Fragen zur Rechnerbenutzung oder zu den Übungsaufgaben wenden Sie sich bitte jederzeit an einen der TutorInnen!**

Aufgaben (insgesamt 0 Punkte - Einstiegsübungen)

0.1 Längenmaße

In den Vereinigten Staaten wird das *angloamerikanische* Maßsystem (English system) verwendet, während in den meisten anderen Ländern das *metrische* System verwendet wird. Die folgende Tabelle gibt Umrechnungsverhältnisse für einige Längeneinheiten an:

English		Metrisch
1 inch		2.54 cm
1 foot	12 in.	
1 yard	3 ft.	
1 rod	5(1/2) yd.	

Tabelle 1: Umrechnungsfaktoren für Längenangaben.

Schreiben Sie Prozeduren für die Umrechnung von Längenangaben.

a) Entwickeln Sie die Umrechnungs-Prozeduren `inches->cm`, `feet->inches`, `yards->feet` sowie `rods->yards`. Binden Sie benötigte Konstanten an Namen.

b) Erstellen Sie Prozeduren, die weitere Einheiten umrechnen: `feet->cm`, `yards->cm` und `rods->inches`. Benutzen Sie dazu nur Funktionen, die Sie bereits definiert haben!

Abgabe: Programm `measurements.rkt`

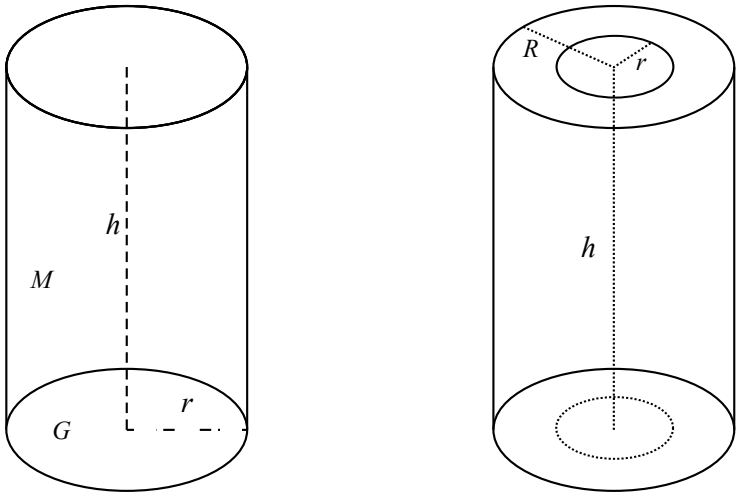
Lösungsvorschlag:

```
;; Die ersten drei Zeilen dieser Datei wurden von DrRacket eingefügt. Sie enthalten Metadaten
;; über die Sprachebene dieser Datei in einer Form, die DrRacket verarbeiten kann.
#reader(lib "DMdA-beginner-reader.ss" "deinprogramm")((modname measurements) (read-case-sensitive #f)
     (teachpacks ()) (deinprogramm-settings #(#f write repeating-decimal #f #t none explicit #f ())))
; conversion factors for the different units of measurement
(define inch->cm-factor 2.54)
(define foot->inch-factor 12)
(define yard->foot-factor 3)
(define rod->yard-factor (+ 5 1/2))

; convert inches to cm
(: inches->cm (real -> real))
(check-within (inches->cm 1) inch->cm-factor 0.0001)
(check-within (inches->cm 2) 5.08 0.0001)

(define inches->cm
  (lambda (l)
    (* l inch->cm-factor)))

; convert feet to inches
(: feet->inches (real -> real))
(check-within (feet->inches 1) foot->inch-factor 0.0001)
(check-within (feet->inches 5) 60 0.0001)

(define feet->inches
  (lambda (l)
    (* l foot->inch-factor)))

; convert yards to feet
(: yards->feet (real -> real))
(check-within (yards->feet 1) yard->foot-factor 0.00001)
(check-within (yards->feet 23) 69 0.00001)

(define yards->feet
  (lambda (l)
    (* l yard->foot-factor)))

; convert rods to yards
(: rods->yards (real -> real))
(check-within (rods->yards 1) rod->yard-factor 0.00001)
(check-within (rods->yards 42) 231 0.00001)

(define rods->yards
  (lambda (l)
    (* l rod->yard-factor)))

; convert feet to cm
(: feet->cm (real -> real))
(check-within (feet->cm 3) 91.44 0.00001)
(check-within (feet->cm 12.4) 377.952 0.0001)

(define feet->cm
  (lambda (l)
    (inches->cm (feet->inches l))))

; convert yards to cm
(: yards->cm (real -> real))
(check-within (yards->cm 254) 23225.76 0.0001)
(check-within (yards->cm 1) 91.44 0.0001)

(define yards->cm
  (lambda (l)
    (inches->cm (feet->inches (yards->feet l)))))

; convert rods to inches
(: rods->inches (real -> real))
(check-within (rods->inches 1) 198 0.000001)
(check-within (rods->inches 23.5) 4653 0.0001)

(define rods->inches
  (lambda (l)
    (feet->inches (yards->feet (rods->yards l)))))
```

Listing 1: Längenmaße - Lösung

0.2 Zylinder

Ein (Kreis-)Zylinder ist definiert durch die Höhe h und den Radius r der Grundfläche G.
Ein Hohlzylinder (auch Rohr, engl. *pipe*) ist definiert durch die Höhe h, den Außenradius R und den Innenradius r.



(a) Zylinder (b) Hohlzylinder

Schreiben Sie Prozeduren, die die Oberfläche von Zylindern und Hohlzylindern berechnen. Dabei sind folgende Teilprobleme zu lösen:

a) Schreiben Sie Prozeduren `circle-area` und `circle-perimeter`, die Fläche und Umfang eines Kreises mit gegebenem Radius r berechnen.

Die Fläche ergibt sich als $A = \pi r^2$, der Umfang als $U = 2\pi r$. Erkennen und abstrahieren Sie Teilprobleme!

b) Schreiben Sie eine Prozedur `cylinder-side-area`, die aus dem Radius r der Grundfläche und der Höhe h die Mantelfläche M eines Zylinders berechnet. Die Mantelfläche bezeichnet die seitliche Fläche, die durch die Grundflächen begrenzt wird.

Schreiben Sie außerdem eine Prozedur `cylinder-area`, die die Oberfläche eines Zylinders berechnet.

c) Schreiben Sie eine Prozedur `pipe-area`, die die Oberfläche eines Hohlzylinders berechnet. Gegeben sind dabei die Höhe h des Hohlzylinders, der Außenradius R und der Innenradius r.

Benutzen Sie dazu Prozeduren, die Sie bereits definiert haben!

Abgabe: Programm `pipe-area.rkt`

Lösungsvorschlag:

```
;; Die ersten drei Zeilen dieser Datei wurden von DrRacket eingefügt. Sie enthalten Metadaten
;; über die Sprachebene dieser Datei in einer Form, die DrRacket verarbeiten kann.
#reader(lib "DMdA-beginner-reader.ss" "deinprogramm")((modname pipe-area) (read-case-sensitive #f)
     (teachpacks ()) (deinprogramm-settings #(#f write repeating-decimal #f #t none explicit #f ())))
(define pi 3.1415)

; compute the square of a number
(: square (number -> number))
(check-expect (square 1) 1)
(check-expect (square 4) 16)
(check-within (square 0.5) 0.25 0.00001)

(define square
  (lambda (x)
    (* x x)))

; compute the area of a circle with radius r
(: circle-area (real -> real))
(check-within (circle-area 5) (* pi (square 5)) 0.00001)
(check-within (circle-area 3.5) 38.483 0.001)

(define circle-area
  (lambda (r)
    (* pi (square r))))

; compute the perimeter of a circle with radius r
(: circle-perimeter (real -> real))
(check-within (circle-perimeter 5) (* 2 pi 5) 0.000001)
(check-within (circle-perimeter 3.5) 21.9905 0.00001)

(define circle-perimeter
  (lambda (r)
    (* 2 pi r)))

; compute the side area of a cylinder with radius r and length l
(: cylinder-side-area (real real -> real))
(check-within (cylinder-side-area 3 8) 150.792 0.001)
(check-within (cylinder-side-area 1.2 7) 52.777 0.001)

(define cylinder-side-area
  (lambda (r l)
    (* l (circle-perimeter r))))

; compute the surface area of a cylinder with radius r and length l
(: cylinder-area (real real -> real))
(check-within (cylinder-area 3.5 8) 252.89 0.001)
(check-within (cylinder-area 23 42) 9393.085 0.000001)

(define cylinder-area
  (lambda (r l)
    (+ (* 2 (circle-area r))
       (cylinder-side-area r l))))

; compute the surface area of a pipe with length l, outer radius R
; and inner radius r (R > r!)
(: pipe-area (real real real -> real))
(check-within (pipe-area 10 4 3) 483.79 0.001)

(define pipe-area
  (lambda (length outer-radius inner-radius)
    (- (+ (cylinder-area outer-radius length)
          (cylinder-side-area inner-radius length))
       (* 2 (circle-area inner-radius)))))
```

Listing 2: Zylinderproblem - Lösung