Grundlagen der Informatik 1

Wintersemester 2009/2010

Prof. Dr. Max Mühlhäuser, Dr. Rößling
http://proffs.tk.informatik.tu-darmstadt.de/teaching

Übung 1 - Lösungsvorschlag Version: 1.0 19. 10. 2009

---

1 Mini Quiz

Kreuzen Sie bei den folgenden Aufgaben die richtige Lösung oder die richtigen Lösungen an.

1. Betrachten Sie die folgende Scheme-Funktion:

```
1 (define (sth x)
2   (if (< x 5) 'x+1 'x*2))
```

   Was wird hier definiert?
   ☒ Eine Prozedur. ☐ Ein Symbol. ☐ Eine Zahl.

   Welches Ergebnis liefert der Aufruf von (sth 4) zurück?
   ☐ 5 ☐ 8 ☒ Ein Symbol. ☐ Einen String.

2. Was bedeutet „Prozedurale Abstraktion“?

   ☒ Die Zerlegung eines großen Problems in kleinere Teilprobleme.
   ☒ Die Auslagerung von Berechnungen in eigene Prozeduren.
   ☐ Abstraktion dient dazu, Details einer Berechnung zu zeigen.

2 Fragen

1. Was sind die Strukturierungsmechanismen einer Programmiersprache? Nennen Sie für jeden Mechanismus in Scheme ein Beispiel.
2. Nennen Sie die generellen Schritte beim Design von Programmen.
3. Wozu werden Symbole in Scheme verwendet? Warum werden nicht stattdessen Strings verwendet?

Lösungsvorschlag:

1. **Strukturierungsmechanismen**

   - Primitive Ausdrücke: z.B. 23, +, false
   - Kombinationsmittel: (Anwendung einer Prozedur) z.B. (+ 3 4), (or true false)
   - Abstraktionsmittel: define, z.B. (define average (a b)...)

2. **Schritte beim Design**

- 1. Verstehen was der Zweck des Programmes ist.
- 2. Programmbeispiele ausdenken.
- 3. Implementierung des Programmkörpers.
- 4. Testen.
- Beispiel: Es ist ein Programm zu schreiben, das anhand von Messwerten überprüft, ob eine industriell gefertigte Ware innerhalb gegebener Toleranzen liegt.
  a) Überlegen, welche Werte überprüft werden müssen, was die maximale Abweichung sein darf und welche Berechnungswege es dafür gibt. Was sollen die Eingaben und Ausgaben der Programms des Programms sein?
  b) Als nächstes folgt die Überlegung für eine grobe Struktur des Programmes. Welche Dinge müssen berechnet werden? Welche sind fest? Wie sollen die Berechnungen für verschiedene konkrete Eingaben aussehen?
  c) Dann Programmierung der einzelnen Prozeduren und Tests der Hilfsprozeduren und des Gesamtprogramms, ob es diese gewünschten Ergebnisse berechnet, insbesondere auch in Grenzfällen.

3. **Symbole** benutzt man für Namen die sich nicht ändern, z.B. Personennamen. Symbole sind atomar wie andere Primitive (Zahlen, boolesche Werte). Das bedeutet, man kann nicht auf ihre einzelnen Bestandteile zugreifen (etwa die Buchstaben in einem Namen). Mit Symbolen ist ein sehr effizienter Vergleich möglich.

   Strings hingegen werden für Textdaten verwendet, die sich ändern oder erst zur Laufzeit feststehen, etwa weil sie während des Programmablaufs eingelesen werden. Bei Strings kann man auf einzelne Bestandteile zugreifen. Die Überprüfung ob zwei Strings identisch sind erfordert eine zeichenweise Überprüfung. Dies ist viel aufwändiger als zwei Symbole zu vergleichen.

3 Scheme-Syntax (K)

3.1 Klammerung

Ergänzen Sie folgende Ausdrücke durch Klammern, so dass jeweils ein gültiger Scheme-Ausdruck entsteht. Werten Sie anschließend diese Ausdrücke aus.

1. `+ + + 7 3 4 * 2 3`
2. `* + 1 9 - 4 + 1 1`
3. `not or true and true false or true false`

Lösungsvorschlag:

1. (+ (+ (+ 7 3) 4) (* 2 3)) = 20
2. (* (+ 1 9) (- 4 (+ 1 1))) = 20
3. (not (or true (and true false) (or true false))) = false

3.2 Präfix-Notation

Übersetzen Sie die folgenden mathematischen Ausdrücke in äquivalente Schemeausdrücke. Die Ausdrücke sollen nicht vereinfacht (gekürzt oder berechnet) werden. 4^3 dürfen Sie durch eine entsprechende Multiplikation ersetzen.

1. $\frac{13-3}{6*3} * (100 - 32)$
2. $\frac{7*2}{28} * \frac{2}{3}$
3. $\frac{4^3}{8} + (4 - 12)$

Lösungsvorschlag:

1. (* (/ (- 13 3) (* 6 3)) (- 100 32))
2. (* (/ (* 7 2) 28) (/ 2 3)) [oder (* (/ (* 7 2) 28) 2/3)]
3. (+ (/ (* 4 4 4) 8) (- 4 12))

4 Tetraedervolumen (K)

In der folgenden Aufgabe soll schrittweise eine Prozedur `volume` in Scheme erstellt werden, die das Volumen eines Tetraeders mit gegebener Kantenlänge berechnet. Dabei soll nach dem Top-Down Ansatz vorgegangen werden.
Hinweis: Das Volumen V eines Tetraeders mit Kantenlänge a berechnet sich als $V(a) = \frac{\sqrt{2}}{12}a^3$, die Funktion zur Berechnung der Quadratwurzel lautet in Scheme $sqrt$.

1. Geben Sie den Vertrag, die Beschreibung und ein Beispiel für die Prozedur `volume` an. `volume` soll einen Wert für die Kantenlänge konsumieren und das zugehörige Tetraedervolumen berechnen.
2. Schreiben Sie nun die Definition der Prozedur in Scheme auf. Gehen Sie hierbei davon aus, dass Sie auf ihrer Wunschliste eine weitere Prozedur `pow3: number -> number` (berechnet x^3 für eine Eingabe x) sowie eine Konstante `k` (für $\frac{\sqrt{2}}{12}$) haben. Geben Sie außerdem einen Test für die Prozedur `volume` an.
3. Erstellen Sie nun die Prozedur `pow3` mit Vertrag, Beschreibung und Beispiel, und definieren Sie die Konstante `k` mit einem Wert von $\frac{\sqrt{2}}{12}$.
4. Überlegen Sie sich, wie Sie bei der Programmierung von `volume` nach dem Bottom-Up Verfahren vorgegangen wären.

Lösungsvorschlag:

```
1 ;; 4.3)
2 ;;
3 ;; Contract: pow3: number -> number
4 ;; Purpose: calculates the cubic value of input a
5 ;; Example: (pow3 4) should be 64
6 (define (pow3 a)
7   (* (* a a) a) ;; Remark: (* a a a) is also possible
8   )
```

```
;; Tests:
(check-expect (pow3 4) 64)
(check-expect (pow3 5) 125)

;; definition of a constant for sqrt(2) / 12
(define k (/ (sqrt 2) 12))

;; 4.1)
;;
;; Contract: volume: number -> number
;; Purpose: Calculates the volume of a tetraeder with side length a
;; Example: (volume 5) should produce 14.73
;;
;; ;; 4.2)

(define (volume a)
  (* k (pow3 a))
)

;; Tests:
(check-within (volume 3) 3.18 0.01)
(check-within (volume 5) 14.73 0.01)
```

5 Auswertung von Ausdrücken (K)

Gehen Sie von folgender Scheme-Prozedur aus:

```
;; Contract: weighted-average : number number number -> number
;; Purpose: to calculate a weighted average of 3 given numbers x,y,z
;;          using following formula = (x+y+y+z)/4
;; Example: (weighted-average 3 6 7) should produce 5.5

;; Definition:
(define (weighted-average x y z)
  (/ (+ x y y z) 4))

;; Test
(check-expect (weighted-average 3 8 1) 5)
(check-within (weighted-average 3 1 4) 2.25 0.01)
```

Überlegen Sie, wie folgende Ausdrücke in applikativer und normaler Auswertungsreihenfolge ausgewertet werden, und schreiben Sie zu jedem Ausdruck eine Variante auf (siehe Folie T1.58ff).

1. (weighted−average (weighted−average 15 10 5) (/ 36 9) (∗ 2 3))
2. (weighted−average (∗ 3 4) (weighted−average 4 13 2) (weighted−average 31 50 13))

Lösungsvorschlag:

a) Applikative Auswertungsreihenfolge:

```
(weighted-average (weighted-average 15 10 5) (/ 36 9) (* 2 3))
(weighted-average (/ (+ 15 10 10 5) 4) (/ 36 9) (* 2 3))
(weighted-average (/ 40 4) (/ 36 9) (* 2 3))
(weighted-average 10 (/ 36 9) (* 2 3))
(weighted-average 10 4 (* 2 3))
(weighted-average 10 4 6)
(/ (+ 10 4 4 6) 4)
(/ 24 4)
6
```

Normale Auswertungsreihenfolge:

```
(weighted-average (weighted-average 15 10 5) (/ 36 9) (* 2 3))
(/ (+ (weighted-average 15 10 5) (/ 36 9) (/ 36 9) (* 2 3)) 4)
(/ (+ (/ (+ 15 10 10 5) 4) (/ 36 9) (/ 36 9) (* 2 3)) 4)
(/ (+ (/ 40 4) (/ 36 9) (/ 36 9) (* 2 3)) 4)
(/ (+ 10 (/ 36 9) (/ 36 9) (* 2 3)) 4)
(/ (+ 10 4 (/ 36 9) (* 2 3)) 4)
(/ (+ 10 4 4 (* 2 3)) 4)
(/ (+ 10 4 4 6) 4)
(/ 24 4)
6
```

b) Applikative Auswertungsreihenfolge:

```
(weighted-average (* 3 4) (weighted-average 4 13 2)
  (weighted-average 31 50 13))
(weighted-average 12 (weighted-average 4 13 2)
  (weighted-average 31 50 13))
(weighted-average 12 (/ (+ 4 13 13 2) 4)
  (weighted-average 31 50 13))
(weighted-average 12 (/ 32 4) (weighted-average 31 50 13))
(weighted-average 12 8 (weighted-average 31 50 13))
(weighted-average 12 8 (/ (+ 31 50 50 13) 4))
(weighted-average 12 8 (/ 144 4))
(weighted-average 12 8 36)
(/ (+ 12 8 8 36) 4)
(/ 64 4)
16
```

Normale Auswertungsreihenfolge:

```
(weighted-average (* 3 4) (weighted-average 4 13 2)
  (weighted-average 31 50 13))
(/ (+ (* 3 4) (weighted-average 4 13 2)
  (weighted-average 4 13 2) (weighted-average 31 50 13)) 4)
(/ (+ (* 3 4) (/ (+ 4 13 13 2) 4) (weighted-average 4 13 2)
  (weighted-average 31 50 13)) 4)
(/ (+ (* 3 4) (/ (+ 4 13 13 2) 4) (/ (+ 4 13 13 2) 4)
  (weighted-average 31 50 13)) 4)
(/ (+ (* 3 4) (/ (+ 4 13 13 2) 4) (/ (+ 4 13 13 2) 4)
  (/ (+ 31 50 50 13) 4)) 4)
(/ (+ (* 3 4) (/ 32 4) (/ (+ 4 13 13 2) 4)
  (/ (+ 31 50 50 13) 4)) 4)
(/ (+ (* 3 4) (/ 32 4) (/ 32 4) (/ (+ 31 50 50 13) 4)) 4)
(/ (+ (* 3 4) (/ 32 4) (/ 32 4) (/ 144 4)) 4)
(/ (+ 12 (/ 32 4) (/ 32 4) (/ 144 4)) 4)
(/ (+ 12 8 (/ 32 4) (/ 144 4)) 4)
(/ (+ 12 8 8 (/ 144 4)) 4)
(/ (+ 12 8 8 36) 4)
(/ 64 4)
16
```

6 Steuern

Die Steuern in Land X ergeben sich aus dem Einkommen multipliziert mit dem Steuersatz. Der Steuersatz ist wiederum 0.5% je tausend Euro Einkommen (abgerundet). So würde ein Einkommen von 40.000 Euro so einem Steuersatz von $0.5\% * 40 = 20\%$ und einer zu zahlenden Steuer von 8.000 Euro entsprechen, ein Einkommen von 23.700 Euro einem Steuersatz von 11,5% etc.
Hinweis: Zum Runden von Werten kann die Funktion `round` verwendet werden. Aber Vorsicht: diese Funktion rundet ab 0.5 auf, siehe auch `http://en.wikipedia.org/wiki/Rounding`!

1. Schreiben Sie eine Funktion `get-taxrate`, die ein Einkommen konsumiert und den Steuersatz (in Prozent, d.h. 40 für 40%) produziert. Schreiben Sie dazu eine Funktion `round-off`, die eine gegebene Zahl abrundet. Geben Sie zu beiden Funktionen auch je Vertrag, Beschreibung, Beispiel und mindestens zwei Tests an.

2. Schreiben Sie eine Funktion `get-income`, die ein Einkommen konsumiert und das Einkommen nach Abzug der Steuer produziert. Geben Sie auch hierzu Vertrag, Beschreibung, Beispiel und mindestens zwei Tests an.

Lösungsvorschlag:

```
;; round-down: number->number
;; rounds off the given number
;; Example: (round-down 3.7) should produce 3
(define (round-down number)
  (if (> (round number) number)
      (- (round number) 1)
      (round number) ))

;; Tests
(check-expect (round-down 3.5) 3)
(check-expect (round-down 4.1) 4)

;; get-taxrate: number -> number
;; returns the taxrate in percent for the given income
;; Example: (get-taxrate 12700) should be 6
(define (get-taxrate income)
  (/(round-down (/ income 1000)) 2))

;; Tests
(check-expect (get-taxrate 23600) 11.5)
(check-expect (get-taxrate 40000) 20)

;; get-income number -> number
;; returns the netto income for a given brutto income
;; Example: (get-income 40000) should be 32000
(define (get-income income)
  (- income (* income (/ (get-taxrate income) 100))))

;; Tests
(check-expect (get-income 23700) 20974.5)
(check-expect (get-income 12400) 11656)
```

Hausübung

Die Vorlagen für die Bearbeitung werden im GdI1-Portal bereitgestellt. Kommentieren Sie Ihren selbst erstellten Code. Die Hausübung muss bis zum Abgabedatum im GdI1-Portal abgegeben werden. Der Fachbereich Informatik misst der Einhaltung der Grundregeln der wissenschaftlichen Ethik großen Wert bei. Zu diesen gehört auch die strikte Verfolgung von Plagiarismus. Mit der Abgabe Ihrer Hausübung bestätigen Sie, dass Sie bzw. Ihre Gruppe alleiniger Autor des gesamten Materials sind. Falls Ihnen die Verwendung von Fremdmaterial gestattet war, so müssen Sie dessen Quellen deutlich zitieren. Falls Sie die Hausübung in einer Lerngruppe bearbeitet haben, geben Sie dies bitte deutlich bei der Abgabe an. Alle anderen Mitglieder der Lerngruppe müssen als Abgabe einen Verweis auf die gemeinsame Bearbeitung einreichen, damit die Abgabe im Portal auch für sie bewertet werden kann.

Abgabedatum: Freitag, 30. 10. 2009, 16:00 Uhr

Denken Sie bitte daran, Ihren Code hinreichend gemäß den Vorgaben zu kommentieren (Scheme: Vertrag, Beschreibung und Beispiel sowie zwei Testfälle pro Funktion; Java: JavaDoc). Zerlegen Sie Ihren Code sinnvoll und versuchen Sie, wo es möglich ist, bestehende Funktionen wiederzuverwenden. Wählen Sie sinnvolle Namen für Hilfsfunktionen und Parameter.

7 Aller Anfang...

Installieren Sie DrScheme von `http://www.drscheme.org` auf Ihrem Rechner oder rufen Sie DrScheme mit dem Befehl 'drscheme' auf den Rechnern der RBG auf. Machen Sie sich mit den Grundfunktionen des Programms vertraut. Einen guten Ausgangspunkt bietet die kurze Tour auf `http://www.plt-scheme.org/software/drscheme/tour/`. Probieren Sie einige der Beispiele aus der Vorlesung aus, z.B. T1.19f., T1.33f, T2.5f., T2.14f. Sie finden diese auch im Portal in der *Datenbank*. *Wichtig*: Stellen Sie das Sprachlevel für diese und die folgenden Aufgaben auf *How To Design Programs -> Anfänger* (bzw. *Beginning Student* in der englischen Version).
Hinweis für die folgenden Aufgaben: Kommentieren Sie **alle** von Ihnen definierten Funktionen mit Vertrag, Beschreibung, Beispiel sowie mindestens zwei Tests. Dies ist ein Bestandteil der Bewertung! Versuchen Sie, Teilprobleme sinnvoll in Hilfsfunktionen auszulagern. Außerdem dürfen Sie Funktionen und Hilfsfunktionen aus anderen Aufgabenteilen wiederverwenden.

8 Hello GdI Portal (1 Punkt)

Die Lösung Ihrer Hausübungen müssen Sie online im GdI Portal einreichen. Gehen Sie daher mit ihrem Browser auf die URL `http://proffs.tk.informatik.tu-darmstadt.de/teaching`.

1. Als erstes sollten Sie gegebenenfalls die eingestellte Sprache anpassen, z.B. auf *Deutsch - Du (de_du)* oder *English*. Die Auswahlmöglichkeiten finden Sie oben rechts auf der Startseite des Portals.

2. Klicken Sie auf den Kurs *Grundlagen der Informatik I, WS 2009/2010*. Falls Sie bereits im Portal registriert sind, geht es weiter bei Schritt 5.

3. Legen Sie ein neues Benutzerkonto an, indem Sie auf den Button „Neuen Zugang anlegen?“ klicken. Füllen Sie das nun erscheinende Formular vollständig aus und achten Sie darauf, eine von Ihnen regelmäßig gelesene Mailadresse zu nutzen. Bitte klicken Sie am Ende auf „Meinen neuen Zugang anlegen (Registrierung)“. Korrigieren Sie ggf. Ihre Einträge, bis das Formular erfolgreich abgesendet werden konnte.

4. An die angegebene Emailadresse wird nun eine Bestätigung geschickt. Bitte öffnen Sie diese Mail und klicken Sie auf den darin enthaltenen Link; es öffnet sich wiederum eine Portalseite. Wechseln Sie nun zur Kursseite.

5. Überprüfen Sie Ihr Nutzerprofil: ist der Name vollständig und korrekt angegeben? Haben Sie Ihre Matrikelnummer angegeben? Ohne diese Angaben können wir für Sie am Ende keine Studienleistung an das Prüfungsamt melden! **Hinweis:** Die Matrikelnummer ist für andere Studierende nicht sichtbar.

6. Neben Foren und den Vorlesungsmaterialien finden Sie im Portal auch Aufgaben. Über diese geben Sie auch Ihre Hausaufgaben ab. Hausübungsabgaben können nur Sie, der Dozent und Ihr Tutor einsehen. Ihr Tutor wird versuchen, Ihre Lösung innerhalb von 7 Tagen nach der Abgabe zu korrigieren. Die Korrektur können Sie dann als Kommentar zu Ihrer Lösung lesen. Je früher Sie ihre Lösung abgeben, desto früher haben Sie in der Regel auch die Korrektur durch den Tutor!

Zusätzlich stehen im Portal die folgenden Hilfsmittel als separate Kurse für Sie bereit:

- Kurs „Hilfen zur Nutzung des Portals“: hier finden Sie Information zum Portal und wie man es möglichst effizient nutzen kann. Das Portal kann sehr viel, und nicht alles davon sieht man auf den ersten Blick. Daher empfehlen wir Ihnen sehr, sich auch in diesen Kurs einzutragen und die dortigen Materialien durchzuarbeiten!

- Kurs „Tipps zum effektiven Studieren“: hier erhalten Sie Tipps rund um das Studium, etwa zum Bilden von Lerngruppen oder der Vorbereitung auf Klausuren. Eine Einschreibung in diesen Kurs ist daher ebenfalls ratsam. Zusätzlich arbeiten wir hier noch an Beiträgen mit Tipps zur Programmierung.

Hinweis: Um den Punkt für diese Aufgabe zu erhalten, müssen Sie sich im *Übungsgruppenforum* Ihrer Übungsgruppe kurz vorstellen. Sie sollten dabei Ihren Namen angeben und sich kurz beschreiben (Hobbys und Interessen), aber *bitte nicht Ihre Matrikelnummer angeben*.
Sollten Sie Probleme mit der Bedienung des Portals haben, sprechen Sie mit Ihren Kommilitonen und probieren Sie, selbstständig eine Lösung zu finden. Sollten Sie absolut nicht weiterkommen, fragen Sie Ihren Tutor. Dabei hilft es, wenn Sie Ihre Frage klar formulieren und aufschreiben. Nur so kann Ihr Tutor Ihnen schnell helfen. Sollte Ihr Tutor Ihnen nicht weiterhelfen können, wenden Sie sich an die Veranstalter.
Hinweis: Bitte geben Sie diese Selbstbeschreibung **vor** der Einreichung der zweiten Hausaufgabe dieses Übungsblatts ab, damit der Tutor sie bei der Korrektur direkt mit berücksichtigen kann!

9 Objekt im Kreis (4 Punkte)

In den folgenden Aufgaben soll überprüft werden, ob sich ein Objekt in einem Kreis befindet.
Hinweis: Es werden keine trigonometrischen Funktionen (sin, cos, tan) zur Lösung dieser Aufgabe benötigt. Nutzen Sie stattdessen die beiden folgenden eingebauten Prozeduren:

- `sqrt: number -> number` berechnet für Eingabe x den Wert $\sqrt{x}$.
- `sqr: number -> number` berechnet für Eingabe x den Wert x^2.

9.1 Punkt im Kreis (2 Punkte)

Gegeben ist ein Kreis, beschrieben durch die x- und y-Koordinaten seines Mittelpunkts und seinen Radius, sowie ein Punkt, beschrieben durch seine x- und y-Koordinate.

1. Geben Sie für die Prozedur `circle-contains-point?` den Vertrag, die Beschreibung und ein Beispiel an. `circle-contains-point?` soll dabei fünf Parameter konsumieren: einen Kreis, beschrieben durch die Koordinaten des Mittelpunkts und dem Kreisradius, sowie die x- und y-Koordinate eines Punkts. Die Prozedur soll `true` produzieren, wenn sich der Punkt innerhalb des Kreises befindet, sonst `false`.

2. Implementieren Sie die Prozedur in Scheme.

9.2 Kreis im Kreis (2 Punkte)

In dieser Aufgabe soll überprüft werden, ob sich ein Kreis vollständig in einem anderen Kreis befindet. Die Kreise werden dabei mit der Position ihrer Mittelpunkte und ihrem Radius angegeben.

1. Geben Sie für die Prozedur `circle-contains-circle?` den Vertrag, die Beschreibung und ein Beispiel an. `circle-contains-circle?` soll dabei sechs Parameter konsumieren: zwei Kreise, beschrieben durch die Koordinaten des Mittelpunkts und der Länge des Radius. Die Prozedur soll `true` oder `false` produzieren und somit berechnen, ob der erste Kreis den zweiten vollständig enthält.

2. Implementieren Sie nun die Prozedur in Scheme.