Grundlagen der Informatik 1
WS 2008/09

Prof. Mühlhäuser, Dr. Rößling, Melanie Hartmann, Daniel Schreiber
http://proffs.tk.informatik.tu-darmstadt.de/gdi1

Übung 1 Lösungsvorschlag v1.0 20.10.2008

---

1 Mini Quiz

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

1. Betrachten Sie die folgende Zeichenfolge:

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

   Was wird hier definiert ?
   ☒ Eine Funktion. ☐ 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 wird die Aufgabe gestellt ein Programm zu schreiben, das anhand von Messwerten überprüft, ob eine industriell gefertigte Ware innerhalb gegebener Toleranzen liegt.*
     - *Überlegung 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?*
     - *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?*
     - *Dann Programmierung der einzelnen Prozeduren und Tests der Hilfsprozeduren und des Gesamtprogramms, ob es diese gewünschten Ergebnisse berechnet, insbesondere auch in Grenzfällen.*

3. **Symbole**
   Symbole benutzt man für Namen die sich nicht ändern, z.B. Personenanmen. Symbole sind atomar wie andere Primitive (Zahlen, Boolsche 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 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. Zu überprüfen 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 und werten Sie diese 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).

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 die 3. Potenz der Eingabe) 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, wie 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
```

```
(define (pow3 a)
  (* (* a a) a) ;; Remark: (* a a a) is also possible
  )
;; Test:
(pow3 4)
;; should be
64

;; 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:
(volume 3)
;; should be
3.18
(volume 5)
;; should be
14.73
```

5 Auswertung von Ausdrücken(K)

Gehen Sie von folgender Scheme-Prozedur aus:

```
;; Contract: AverageW : 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: (AverageW 3 6 7) should produce 5.5

;; Definition:
(define (AverageW x y z)
  (/ (+ x y y z) 4))

;; Test
(AverageW 3 8 1)
;; should be
5
```

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

1. `(AverageW 2 17 24)`
2. `(AverageW (+ 3 4) (AverageW 15 7 22) -5)`

Lösungsvorschlag:

a) Hier gibt es keinen Unterschied zwischen den beiden Auswertungsreihenfolgen.

```
(AverageW 2 17 24)
(/ (+ 2 17 17 24) 4)
(/ 60 4)
60/4
15
```

b) Applikative Auswertungsreihenfolge:

```
(AverageW (+ 3 4) (AverageW 15 7 22) -5)
(AverageW 7 (AverageW 15 7 22) -5)
(AverageW 7 (/ (+ 15 7 7 22) 4) -5)
(AverageW 7 (/ 51 4) -5)
(AverageW 7 51/4 -5)
(/ (+ 7 51/4 51/4 -5) 4)
(/ 55/2 4)
55/8
6.875
```

Normale Auswertungsreihenfolge:

```
(AverageW (+ 3 4) (AverageW 15 7 22) -5)
(AverageW (+ 3 4) (AverageW 15 7 22) -5)
(/ (+ (+ 3 4) (AverageW 15 7 22) (AverageW 15 7 22) -5) 4)
(/ (+ (+ 3 4) (/ (+ 15 7 7 22) 4) (AverageW 15 7 22) -5) 4)
(/ (+ (+ 3 4) (/ (+ 15 7 7 22) 4) (/ (+ 15 7 7 22) 4) -5) 4)
(/ (+ (+ 3 4) (/ 51 4) (/ (+ 15 7 7 22) 4) -5) 4)
(/ (+ (+ 3 4) (/ 51 4) (/ 51 4) -5) 4)
(/ (+ 7 (/ 51 4) (/ 51 4) -5) 4)
(/ (+ 7 51/4 (/ 51 4) -5) 4)
(/ (+ 7 51/4 51/4 -5) 4)
(/ 55/2 4)
55/8
6.875
```

6 Steuern

Die Steuern in Land X berechnen sich nach folgendem einfachen Muster: Die Steuern die man zahlen muss ergeben sich aus dem Einkommen multipliziert mit dem Steuersatz. Der Steuersatz ist wiederum $1/2\%$ je tausend Euro Einkommen (abgerundet). Zum Beispiel, würde ein Einkommen von 40.000 Euro so einem Steuersatz von $1/2\% * 40 = 20\%$ 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: die Funktion rundet ab 0,5 auf!

1. Schreiben Sie eine Funktion `get-taxrate`, die ein Einkommen entgegennimmt und den Steuersatz (in Prozent, d.h. 40 für 40%) zurückgibt. 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 entgegennimmt und das Einkommen nach Abzug der Steuer zurückgibt. Geben Sie auch hierzu Vertrag, Beschreibung, Beispiel und mindestens zwei Tests an.

Lösungsvorschlag:

```
;; Contract: round-off: number->number
;; Purpose: rounds off the given number
;; Example: (round-off 3.7) should produce 3
(define (round-off number)
  (if (> (round number) number) (- (round number) 1) (round number) ))
;; Test
(round-off 3.5)
;; should be
3
(round-off 4.1)
;;should be
4

;; Contract: get-taxrate: number -> number
;; Purpose: returns the taxrate in percent for the given income
;; Example: (get-taxrate 12700) should be 6
(define (get-taxrate income)
  (/(round-off (/ income 1000)) 2))
;; Test
(get-taxrate 23600)
;;should be
11.5
(get-taxrate 40000)
;;should be
20

;; Contract: get-income number -> number
;; Purpose: 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))))
;; Test
(get-income 23700)
;; should be
20974.5
(get-income 12400)
;; should be
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.

Abgabedatum: Fr., 31.10.08, 16:00 Uhr

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 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.
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 mindestens 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/ICS Portal (2P)

Die Lösung Ihrer Hausübungen müssen Sie online im GdI/ICS Portal einreichen. Gehen Sie daher mit ihrem Browser auf die URL http://proffs.tk.informatik.tu-darmstadt.de/gdi1.

1. Melden Sie sich mit Ihrem RBG-Benutzernamen und Passwort auf der Seite an. Sollten Sie keinen RBG Benutzernamen haben, melden Sie sich bei den Veranstaltern.

2. Klicken Sie nun auf den Menüpunkt Übungsgruppen. Sie sehen nun eine Liste aller Übungsgruppen zur Veranstaltung.

3. Finden Sie die Übungsgruppe zu der Sie über WebReg zugeteilt wurden und klicken Sie dort auf Mitgliedschaft anfordern.

4. Ihr Tutor erhält nun eine Nachricht, dass Sie dieser Gruppe beitreten wollen. Er wird Sie nach spätestens einem Tag freischalten. Während Sie auf die Freischaltung warten, können Sie bereits das GdI/ICS Portal erkunden.

5. Sind Sie durch den Tutor in die Übungsgruppe aufgenommen worden, können Sie Lösungen abgeben. Sie müssen dazu auf den Menüpunkt Hausübungsabgabe klicken. Hier können Sie jetzt ihre Lösung eintippen oder auch Dateien mit Scheme Code anhängen. Hausübungsabgaben können nur Sie und ihr Tutor einsehen. Ihr Tutor wird ihre Lösung innerhalb von 7 Tagen nach der Abgabe korrigieren. Die Korrektur können Sie dann als Kommentar zu Ihrer Lösung lesen. Je früher Sie ihre Lösung abgeben, desto früher haben Sie auch die Korrektur durch den Tutor!

6. Sie können auch allgemeine Nachrichten an die Übungsgruppe schreiben. Dies geht unter dem Menüpunkt Gruppenbeitrag. Diese können alle Mitglieder der Übungsgruppe lesen. Nutzen Sie diese Funktion nun um sich kurz mit ihrem Namen in der Gruppe vorzustellen.

Sollten Sie Probleme mit der Bedienung des Portals haben, sprechen Sie mit Ihren Kommilitonen und probieren Sie, selbständig 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.

9 Geraden (4P)

Eine Gerade ist über die Achsenabschnitts-Gleichung $y = mx + n$ definiert. Der Wert m ist dabei die Steigung und n ist der y-Achsenabschnitt der Geraden am Punkt $x = 0$. Sie sollen im Folgenden feststellen, wie sich zwei Geraden zueinander verhalten, das heißt ob sie parallel zueinander liegen oder wo sie sich schneiden. Eine Gerade ist dabei durch zwei Punkte (x_1, y_1) und (x_2, y_2) definiert.

1. Dazu müssen Sie zuerst die Parameter einer Geraden errechnen, die durch zwei Punkte definiert wird:
   - Schreiben Sie eine Funktion `get-m`, die zwei Punkte konsumiert und die Steigung einer Geraden durch diese Punkte errechnet. Verwenden sie dazu die die Formel $m = \frac{y2-y1}{x2-x1}$.
   - Schreiben Sie eine Funktion `get-n`, die zwei Punkte konsumiert und den y-Achsenabschnitt der zugehörigen Geraden ermittelt ($n = y - mx$).
2. Schreiben Sie eine Funktion `is-parallel`, die vier Punkte konsumiert und überprüft, ob die Gerade, die durch die ersten beiden Punkte definiert wird parallel ist zu der Geraden, die durch die letzten beiden Punkte definiert wird. Die Rückgabe soll ein booldescher Wert sein.
3. Schreiben Sie eine Funktion `get-intersection-point-x` und eine Funktion `get-intersection-point-y`, die je vier Punkte konsumieren. Die Funktion soll den x- bzw. den y-Wert des Schnittpunktes der beiden Geraden zurückgeben, die durch die vier Punkte definiert sind (die ersten beiden Punkte definieren die erste Gerade, die letzten beiden die zweite). Verwenden Sie dabei folgende Formeln (m_1, n_1, m_2, n_2 sind dabei die Steigung und Achsenschnittpunkte der ersten bzw. zweiten Geraden): $x = \frac{n_2-n_1}{m_1-m_2}$ und $y = m_1 * x + n_1$.

Lösungsvorschlag:

```
1  ;;Contract: get-m: number number number number -> number
2  ;;Purpose: Given X1 Y1 X2 Y2, the procedure calculates the gradient of a
     line through the points (X1 Y1) and (X2 Y2)
3  ;;Example: (get-m 0 0 2 3) should produce 1.5
4  (define (get-m x1 y1 x2 y2)
5   (/ (- y2 y1) (- x2 x1)))
6  ;; Test
7  (get-m 1 1 3 7)
8  ;; should be
9  3
10 ;; Test
11 (get-m 0 0 3.5 -3.5)
12 ;; should be
13 -1
14
15 ;;Contract: get-n: number number number number -> number
16 ;;Purpose: Given X1 Y1 X2 Y2, the procedure calculates the Y-axis
     intercept of a line through the points (X1 Y1) and (X2 Y2)
17 ;;Example: (get-n 0 0 2 3) should produce 0
18 (define (get-n x1 y1 x2 y2)
19  (- y1 (* (get-m x1 y1 x2 y2) x1)))
20 ;; Test
21 (get-n 1 1 3 7)
22 ;; should be
23 -2
24 ;; Test
```

```
(get-n 0 0 3.5 -3.5)
;; should be
0

;;Contract: is-parallel: number number number number number number number number -> boolean
;;Purpose: Given X1 Y1 X2 Y2 X3 Y3 X4 Y4, the procedure calculates whether the line l1 through the points (X1 Y1) and (X2 Y2) and the line l2 through the points (X3 Y3) and (X4 Y4) are parallel
;;Example: (is-parallel 0 0 2 3 0 0 3.5 -3.5) should produce false
(define (is-parallel x1 y1 x2 y2 x3 y3 x4 y4)
 (= (get-m x1 y1 x2 y2) (get-m  x3 y3 x4 y4)))
;; Test
(is-parallel 1 1 3 7 0 0 3.5 -3.5)
;; should be
false
;; Test
(is-parallel 0 0 1 1 2 2 3 3)
;; should be
true

;;Contract: get-intersection-point-x: number number number number number number number number -> number
;;Purpose: Given X1 Y1 X2 Y2 X3 Y3 X4 Y4, the procedure calculates the x-value of the intersection point of the line l1 through the points (X1 Y1) and (X2 Y2) and the line l2 through the points (X3 Y3) and (X4 Y4). If the lines are parallel an error occurs.
;;Example: (get-intersection-point-x 0 0 2 3 0 0 3.5 -3.5) should produce 0
(define (get-intersection-point-x x1 y1 x2 y2 x3 y3 x4 y4)
(/ (- (get-n x3 y3 x4 y4) (get-n x1 y1 x2 y2)) (- (get-m x3 y3 x4 y4) (get-m x1 y1 x2 y2))))
;; Test
(get-intersection-point-x 1 1 3 7 0 0 3.5 -3.5)
;; should be
0.5
;; Test
(get-intersection-point-x 0 0 1 2 2 2 3 3)
;; should be
0

;;Contract: get-intersection-point-y: number number number number number number number number -> number
;;Purpose: Given X1 Y1 X2 Y2 X3 Y3 X4 Y4, the procedure calculates the y-value of the intersection point of the line l1 through the points (X1 Y1) and (X2 Y2) and the line l2 through the points (X3 Y3) and (X4 Y4). If the lines are parallel an error occurs.
;;Example: (get-intersection-point-y 0 0 2 3 0 0 3.5 -3.5) should produce 0
(define (get-intersection-point-y x1 y1 x2 y2 x3 y3 x4 y4)
(+ (* (get-intersection-point-x x1 y1 x2 y2 x3 y3 x4 y4) (get-m x1 y1 x2 y2)) (get-n x1 y1 x2 y2)))
;; Test
(get-intersection-point-y 1 1 3 7 0 0 3.5 -3.5)
;; should be
-3.5
;; Test
(get-intersection-point-y 0 0 1 2 2 2 3 3)
```

```
;; should be
0
```

10 Flächenberechnung (4P)(K)

Ein Kreis wird durch seinen Radius r und ein Quadrat durch die Länge einer Kante a beschrieben.

1. Definieren Sie eine Funktion `get-circle-area` und eine Funktion `get-square-area`, die r bzw. a konsumiert und die Fläche des zugehörigen Kreises bzw. des Quadrats zurückgibt.
   Hinweis: Die Formeln zur Berechnung der Flächen lauten wie folgt:
   Kreis: $\pi * r^2$, π können Sie durch 3,14 approximieren
   Quadrat: a^2

2. Definieren Sie zwei Funktionen `cut-circles` bzw. `cut-squares`, die entweder zwei Kreise (in Form ihrer Radien, d.h. (cut-circles r_1 r_2)) oder zwei Quadrate (in Form ihrer Seitenlängen, d.h. (cut-squares a_1 a_2)) entgegennimmt. Je nachdem welcher/s der beiden größer ist, soll der/das kkleinere aus dem größeren herausgeschnitten werden. Die Funktion soll die Restfläche des Schnitts zurückliefern.

Hinweis: Alternativ können Sie auch mit structs arbeiten, d.h. nur eine Funktion `cut` und eine Funktion `get-area` definieren, die je nach Typ des übergebenen Objekts bzw. der übergebenen Objekte entscheidet, wie die Fläche bzw. der Schnitt zu berechnen ist.

Lösungsvorschlag:

```
;;define PI
(define p 3.14)

;;Contract: get-circle-area: number -> number
;;Purpose: Given the radius of a circle r, the procedure returns that
  circle's area.
;;Example: (get-circle-area 3) should produce 28.26
(define (get-circle-area r)
(* r r p))
;; Test
(get-circle-area 3.5)
;; should be
38.465
;; Test
(get-circle-area  1)
;; should be
3.14

;;Contract: get-square-area: number -> number
;;Purpose: Given the edge length of a square a, the procedure returns
  that square's area.
;;Example: (get-square-area 3) should produce 9
(define (get-square-area a)
(* a a))
;; Test
(get-square-area 3.5)
;; should be
12.25
```

```
;; Test
(get-square-area  1)
;; should be
1

;;Contract: cut-squares: number number -> number
;;Purpose: Given two squares, the procedure determines the remaining area
   when cutting the smaller from the larger.
;;Example: (cut-squares 5 3) should produce 16
(define (cut-squares a1 a2)
(if (< a1 a2) (- (get-square-area a2) (get-square-area a1)) (- (get-
  square-area a1) (get-square-area a2))))
;; Test
(cut-squares 3 7)
;; should be
40
;; Test
(cut-squares 6 2)
;; should be
32

;;Contract: cut-circles: number number -> number
;;Purpose: Given two circles, the procedure determines the remaining area
   when cutting the smaller from the larger.
;;Example: (cut-circles 5 3) should produce 50.24
(define (cut-circles r1 r2)
(if (< r1 r2) (- (get-circle-area r2) (get-circle-area r1)) (- (get-
  circle-area r1) (get-circle-area r2))))
;; Test
(cut-circles 2 6)
;; should be
100.48
;; Test
(cut-circles 7 3)
;; should be
125.6

;;--------------- Alternative Solution with structs
  ------------------------

(define-struct circle (r))
(define-struct square (a))

;;Contract: get-area: circle-or-square -> number
;;Purpose: Given a circle or a square, the procedure returns its area.
;;Example: (get-area (make-circle 3)) should produce 28.26
(define (get-area object)
  (cond
    [(circle? object) (* (circle-r object) (circle-r object) p)]
    [(square? object) (* (square-a object) (square-a object))]
    )
  )

;; Test
(get-area (make-square 3.5))
;; should be
12.25
;; Test
```

```
80  (get-area  (make-square 1))
81  ;; should be
82  1
83  ;; Test
84  (get-area  (make-circle 1))
85  ;; should be
86  3.14
87
88  ;;Contract: cut: circle-or-square circle-or-square -> number-or-'
      erroneousInput
89  ;;Purpose: Given two circles or two squares, the procedure determines the
       remaining area when cutting the smaller from the larger.
90  ;;          If the two given objects are not of the same type '
      erroneousInput is returned
91  ;;Example: (cut (make-circle 5) (make-circle 3)) should produce 50.24
92  (define (cut object1 object2)
93      ;;test whether both objects are circles or squares
94      (if(or (and (circle? object1) (circle? object2) )  (and (square?
          object1) (square? object2) ))
95          (if (< (get-area object1) (get-area object2))
96              (- (get-area object2) (get-area object1)) (- (get-area
                object1) (get-area object2)))
97        'erroneousInput))
98  ;; Test
99  (cut (make-square 3) (make-square 7))
100 ;; should be
101 40
102 ;; Test
103 (cut (make-circle 7) (make-circle 3))
104 ;; should be
105 125.6
106 ;; Test
107 (cut (make-circle 3) (make-square 5))
108 ;;should be
109 'erroneousInput
```