



Institut für Informatik
Prof. Dr. Peter Thiemann
Jochen Walter

Georges-Köhler-Allee 79
D-79110 Freiburg i. Br.

Freiburg, den 13. November 2000

Informatik 1, WiSe 2000/2001

Übungsblatt 1

Bearbeitung bis zum 2.11.2000

Die Aufgaben auf diesem Blatt können in Teams bearbeitet werden. Bevor drscheme gestartet werden kann, muß setup lang eingegeben werden. Die Aufgaben sind mit einem bis drei Sternen versehen, wobei die Aufgaben mit einem Stern am einfachsten, die mit dreien am schwierigsten sind.

Aufgabe 1 (*):

Wandeln Sie folgende Zahlen in das Dezimalsystem um: 101011_2 , 76_8 , AF_{16} .

Lösung zu Aufgabe 1:

- (a) $101011_2 = 1 \cdot 2^5 + 1 \cdot 2^3 + 1 \cdot 2^1 + 1 \cdot 2^0 = 32 + 8 + 2 + 1 = 43$
- (b) $76_8 = 7 \cdot 8^1 + 6 = 56 + 6 = 62$
- (c) $AF_{16} = 10 \cdot 16 + 15 = 175$

(Zahlen ohne Indizes sind dabei Dezimalzahlen.)

Aufgabe 2 (*):

Geben Sie die Potenzmenge $\mathcal{P}(M)$ der Menge $M = \{1, 2, 3\}$ an.

Lösung zu Aufgabe 2: $\mathcal{P}(\{1, 2, 3\}) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}\}$

Aufgabe 3 (**-***):

- (a) (**) Beweisen Sie das erste Distributivgesetz $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$. indem sie ausrechnen, daß der Ausdruck für jede Variablenbelegung wahr ist. Das heißt also, daß Sie die Wahrheitstafel für den gegebenen Ausdruck aufstellen müssen.
- (b) (***) Beweisen Sie das Idempotenzgesetz der Konjunktion $a \wedge a = a$ mit Hilfe der in der Vorlesung angegebenen Axiome der Booleschen Algebra (nicht durch Aufstellen der Wahrheitstafel).
- (c) (**) Beweisen Sie das erste von DeMorgans Gesetzen $\neg(a \vee b) = \neg a \wedge \neg b$ sowie das Dualitätsprinzip $\neg(\neg a) = a$ durch Aufstellen der Wahrheitstafel.
Zeigen Sie mit diesen Gesetzen, daß man auf die Disjunktion \vee verzichten kann, da man sie durch die Konjunktion \wedge ausdrücken kann.

Lösung zu Aufgabe 3:

- (a) $a = 0, b = 0, c = 0: a \vee (b \wedge c) = 0 \vee (0 \wedge 0) = 0 \vee 0 = 0$
 $(a \vee b) \wedge (a \vee c) = (0 \vee 0) \wedge (0 \vee 0) = 0 \wedge 0 = 0$
 $a = 0, b = 0, c = L: a \vee (b \wedge c) = 0 \vee (0 \wedge L) = 0 \vee 0 = 0$
 $(a \vee b) \wedge (a \vee c) = (0 \vee 0) \wedge (0 \vee L) = 0 \wedge L = 0$
 $a = 0, b = L, c = 0: a \vee (b \wedge c) = 0 \vee (L \wedge 0) = 0 \vee 0 = 0$
 $(a \vee b) \wedge (a \vee c) = (0 \vee L) \wedge (0 \vee 0) = L \wedge 0 = 0$
 $a = 0, b = L, c = L: a \vee (b \wedge c) = 0 \vee (L \wedge L) = 0 \vee L = L$
 $(a \vee b) \wedge (a \vee c) = (0 \vee L) \wedge (0 \vee L) = L \wedge L = L$
 $a = L, b = 0, c = 0: a \vee (b \wedge c) = L \vee (0 \wedge 0) = L \vee 0 = L$
 $(a \vee b) \wedge (a \vee c) = (L \vee 0) \wedge (L \vee 0) = L \wedge L = L$
 $a = L, b = 0, c = L: a \vee (b \wedge c) = L \vee (0 \wedge L) = L \vee 0 = L$
 $(a \vee b) \wedge (a \vee c) = (L \vee 0) \wedge (L \vee L) = L \wedge L = L$
 $a = L, b = L, c = 0: a \vee (b \wedge c) = L \vee (L \wedge 0) = L \vee 0 = L$
 $(a \vee b) \wedge (a \vee c) = (L \vee L) \wedge (L \vee 0) = L \wedge L = L$
 $a = L, b = L, c = L: a \vee (b \wedge c) = L \vee (L \wedge L) = L \vee L = L$
 $(a \vee b) \wedge (a \vee c) = (L \vee L) \wedge (L \vee L) = L \wedge L = L$

(Die Teilergebnisse kann man in den Wahrheitstafeln der Konjunktion (\wedge) und Disjunktion (\vee) ablesen.

- (b) Zunächst gilt wegen des Identitätsgesetzes der Disjunktion $a \wedge a = (a \wedge a) \vee 0$. Danach erhält man durch Anwendung des Komplementaritätsgesetzes der Konjunktion $(a \wedge a) \vee 0 = (a \wedge a) \vee (a \wedge \neg a)$ und mit einem der Distributivgesetze $(a \wedge a) \vee (a \wedge \neg a) = a \wedge (a \vee \neg a)$. Anwendung des Komplementaritätsgesetzes der Disjunktion ergibt $a \wedge (a \vee \neg a) = a \wedge L$, woraus man mit dem Identitätsgesetz der Konjunktion $a \wedge L = a$ erhält.

- (c) Die Wahrheitstafel für das DeMorgansche Gesetz sieht wie folgt aus:

a	b	$(a \vee b)$	$\neg(a \vee b)$	$\neg a$	$\neg b$	$\neg a \wedge \neg b$	$\neg(a \vee b) = \neg a \wedge \neg b$
0	0	0	L	L	L	L	L
0	L	L	0	L	0	0	L
L	0	L	0	0	L	0	L
L	L	L	0	0	0	0	L

Die Wahrheitstabelle für das Dualitätssprinzip:

a	$\neg(\neg a)$
0	0
L	L

Durch Anwendung dieser beiden Gesetze erhält man $a \vee b = \neg(\neg(a \vee b)) = \neg(\neg a \wedge \neg b)$.

Aufgabe 4 (Programmierung, *-*):

- (a) (*) Melden Sie sich auf einem der Pool-Rechner an.
 (b) (**) Ändern Sie Ihr Paßwort. (Merken Sie es sich gut und schreiben Sie es nicht auf!)

- (c) (**) Starten Sie einen Web-Browser wie Netscape. Suchen Sie die Web-Seiten zur Vorlesung und lesen Sie sie durch. Diese Web-Seiten werden in Zukunft als bekannt vorausgesetzt.
- (d) (*) Zeigen Sie das Verzeichnis an, in dem Sie sich gerade befinden.
- (e) (**) Legen Sie eine Textdatei an. Verwenden Sie dazu einen Texteditor wie `xemacs`.
- (f) (**) Kopieren, verschieben und löschen Sie diese Datei.
- (g) (*) Verschaffen Sie sich eine kurze Anleitung zu `xemacs`. Verwenden Sie dazu das Kommando `man xemacs`.
- (h) (*) Melden Sie sich wieder vom Rechner ab.

Lösung zu Aufgabe 4:

- (a) Die Anmeldung erfolgt durch Eingabe von Benutzername und Paßwort beim Login-Bildschirmmaske.
- (b) Die Änderung des Paßwortes geschieht mit dem Kommando `passwd` oder `yppasswd`. Um dieses Kommando eingeben zu können benötigt man ein `xterm`-Fenster. Wenn solch ein Fenster nicht sowieso auf dem Bildschirm zu sehen ist, kann man meist eines über die Button-Leiste öffnen. Es dauert einige Zeit, bis das neue Paßwort gültig ist. Bis zu diesem Zeitpunkt sollte man das Paßwort nicht erneut ändern.
- (c) Die Vorlesungsseiten findet man unter <http://www.informatik.uni-freiburg.de/proglang/teaching/2000-2001info1/>. Man kann sich aber auch von den Seiten des Instituts für Informatik über das Vorlesungsverzeichnis durchklicken.
- (d) Das geschieht durch Eingabe des kommandos `ls`, über dessen zahlreiche Optionen die Manual Pages Auskunft geben.
- (e) Der Aufruf von `xemacs` erfolgt durch `xemacs &`. (Der Ampersand am Ende gibt an, daß der kommandozeileninterpreter nicht auf die Beendigung von `xemacs` wartet, sondern sofort weitere Kommandos akzeptiert.) Für die Verwendung von `xemacs` gibt es eine umfangreiche Online-Dokumentation.
- (f) Zum Kopieren, Verschieben und Löschen verwendet man die Befehle `cp`, `mv` und `rm`.
- (g) Hierzu ist keine Erklärung nötig.
- (h) Das Abmelden geschieht typischerweise über die Button-Leiste oder über ein Pop-Up-Menü, das sich öffnet, wenn man eine Maustaste über dem Bildschirmhintergrund betätigt.

Aufgabe 5 (Programmierung, *-**):

- (a) (*) Wählen Sie einen Mail-Reader. Falls Sie sich mit Mail-Readern nicht auskennen, verwenden Sie `pine`.
- (b) (**) Schicken Sie eine e-mail an Ihren Tutor.
- (c) (**) Lesen Sie dessen Antwort-e-mail.
- (d) (**) Tragen Sie sich in die zur Vorlesung gehörige Mailing-List ein.
- (e) (**) Lesen und interpretieren Sie die Antwort von der Verwaltungssoftware der Mailing-List.

Lösung zu Aufgabe 5: Zu dieser Aufgabe gibt es keine Musterlösung.

Aufgabe 6 (Programmierung, *-*):**

- (a) (*) Starten Sie `drscheme`.
- (b) (**) Verwenden Sie das in DrScheme eingebaute Hilfesystem, um herauszufinden, wieviele Spiele in DrScheme enthalten sind. Welche davon funktionieren nicht?
- (c) (*) Laden Sie den Scheme-Code aus der Vorlesung am Dienstag und führen Sie ihn aus.
- (d) (***) Wieviel Blech benötigt man, um eine zylinderförmige, geschlossene Dose mit einer gegebenen Höhe und einem gegebenen Grundflächenradius herzustellen?
Schreiben Sie dazu eine Scheme-Funktion `dosen-oberflaeche`, die als Argumente zwei Zahlen akzeptiert und als Ausgabe eine weitere Zahl zurückliefert. Das erste Argument ist der Radius, das zweite die Höhe der Dose.
Verwenden Sie hierfür zwei Hilfsfunktionen: Die Hilfsfunktion `dosen-grundflaeche` berechnet aus dem Radius die Grundflaeche der Dose, während die Hilfsfunktion `dosen-mantelflaeche` aus dem Radius und der Höhe der Dose ihre Mantelfläche berechnet.
Speichern Sie Ihre Funktion in einer Datei ab.
- (e) (**) Verwenden Sie den in DrScheme eingebauten Singlestepper, um die Abarbeitung Ihrer Funktion nachzuvollziehen. Setzen Sie dafür den Sprachumfang von DrScheme auf „Beginning Student“.

Lösung zu Aufgabe 6:

- (a) Der Start von DrScheme erfolgt durch

```
setup lang
drscheme&
```
- (b) Die Online-Dokumentation von DrScheme ist nach Themen gegliedert. Wenn man allerdings genau weiß, wonach man sucht, findet man Informationen schneller über die Suchfunktion. Ein geeignetes Stichwort ist *games*. Die Spiele lassen sich direkt aus dem Hilfesystem starten. Es gibt neun verschiedene Spiele, von denen eines (*Paint by numbers*) nicht funktioniert.
- (c) Dazu muß man den Code zuerst mit Hilfe eines Web-Browsers in eine Datei speichern. Danach kann man die Datei mit mit „File→Open“ in DrScheme laden. Unter „Language→Choose Language“ sollte man nun den Sprachumfang auf „Advanced Student“ einstellen. Der Code wird durch Klick auf den „Execute“-Button ausgeführt.
- (d) Eine mögliche Lösung sieht so aus:

```
;;; SIGNATUR
;;; dosen-oberflaeche: number number -> number
;;; ERKLÄRUNG
;;; (dosen-oberflaeche radius hoehe) berechnet die Oberflaeche einer
;;; zylinderförmigen Dose, wobei radius der Radius der Grundfläche
```

```

;;; und hoehe die Höhe der Dose ist.
;;; BEISPIEL
;;; (dosen-oberflaeche 1 2)
;;; => #i18.84955592153876
;;; (Der Präfix #i gibt an, daß die Zahl nicht exakt, d.h. in
;;; Gleitkommadarstellung ist.)
;;; DEFINITION
(define dosen-oberflaeche
  (lambda (radius hoehe)
    (+ (* 2 (dosen-grundflaeche radius))
       (dosen-mantelflaeche radius hoehe))))

;;; SIGNATUR
;;; dosen-grundflaeche: number -> number
;;; ERKLÄRUNG
;;; (dosen-grundflaeche r) berechnet den Flächeninhalt einer (kreisförmigen)
;;; Dose mit Radius r.
;;; BEISPIEL
;;; (dosen-grundflaeche 1)
;;; => #i3.141592653589793
;;; DEFINITION
(define dosen-grundflaeche
  (lambda (r)
    (* pi r r)))

;;; SIGNATUR
;;; dosen-mantelflaeche: number number -> number
;;; ERKLÄRUNG
;;; (dosen-mantelflaeche r h) berechnet den Flächeninhalt des Mantels einer
;;; Dose mit Radius r und Höhe h.
;;; BEISPIEL
;;; (dosen-mantelflaeche 1 2)
;;; => #i12.566370614359172
;;; DEFINITION
(define dosen-mantelflaeche
  (lambda (r h)
    (* 2 pi r h)))

(dosen-oberflaeche 1 2)

```

- (e) Um den Singlestepper einsetzen zu können, muß der in Einzelschritten auszuwertende Ausdruck im Texteditor-Fenster eingegeben werden. Danach startet man den Singlestepper durch Klick auf den *Step*-Button.

Aufgabe 7 (Programmierung, ***):

Schreiben Sie eine sechsstellige Scheme-Funktion `note`, die aus den Punktzahlen der Tests, der Programmierprojekte und der Klausuren die Endnote für die Vorlesung Informatik 1 berechnet.

Lösung zu Aufgabe 7:

```
; $Id: ub1-7.scm,v 1.4 2000/11/13 09:45:06 walterj Exp $
; (Mit Bugfixes von Andreas Witzel)

;;; SIGNATUR
;;; note: number number number number number number -> number
;;; ERKLÄRUNG
;;; (note test1 test2 projekt1 projekt2 klausur1 klausur2) berechnet die
;;; Endnote aus den Punktzahlen der beiden Tests (test1 und test2),
;;; der beiden Programmierprojekte (projekt1 und projekt2) sowie der
;;; beiden Klausuren (klausur1 und klausur2).
;;; Wir treffen die Konvention, daß bei einer Täuschung der Wert -1 übergeben
;;; wird.
;;; Falls die betreffende Person nicht zur Abschlußklausur zugelassen ist,
;;; wird 5.0 zurückgegeben. In diesem Fall kann man aber eigentlich gar nicht
;;; an der Abschlußklausur teilnehmen und bekommt demzufolge dafür auch keine
;;; Punkte die man der Funktion als Argument übergeben könnte. Da in diesem Fall
;;; aber sowieso konstant 5.0 zurückgegeben wird, macht das nichts.
;;; BEISPIEL
;;; (note 10 0 15 16 20 20)
;;; => 2.7
;;; DEFINITION
(define note
  (lambda (test1 test2 projekt1 projekt2 klausur1 klausur2)
    (if (zugelassen test1 test2 projekt1 projekt2 klausur1)
        (let ((gesamt (+ (tests-projekte-punkte test1 test2 projekt1 projekt2)
                        (if (< klausur1 0) 0 klausur1)
                        (if (< klausur2 0) 0 klausur2))))
          (punkte->note gesamt))
        (punkte->note 0))))

;;; SIGNATUR
;;; zugelassen: number number number number number -> boolean
;;; ERKLÄRUNG
;;; (zugelassen test1 test2 projekt1 projekt2 klausur1) überprüft,
;;; ob der Student mit den als Argumente übergebenen Punktzahlen zu der
;;; Abschlußklausur zugelassen ist.
;;; Der Rückgabewert ist #t, falls der Student zur Abschlußklausur
;;; zugelassen ist, #f sonst.
```



```

;;; SIGNATUR
;;; tests-projekte-punkte: number number number number -> number
;;; ERKLÄRUNG
;;; (tests-projekte-punkte test1 test2 projekt1 projekt2) berechnet aus den
;;; Punkten der beiden Tests und der beiden Programmierprojekte die Anzahl
;;; der in Tests und Projekten erworbenen Punkte, die in die Note eingeht.
;;; Die Grundidee für diese Funktion besteht darin, daß zunächst die erste
;;; in die Note eingehende Punktzahl bestimmt wird (entweder das Maximum aller
;;; Punkte oder 0, falls ein Täuschungsversuch vorlag) und die Entscheidung
;;; über die andere Punktzahl an die Hilfsfunktion tests-projekte-punkte-1
;;; übergeben wird.
;;; BEISPIEL
;;; (tests-projekte-punkte -1 15 13 14)
;;; => 15
;;; DEFINITION
(define tests-projekte-punkte
  (lambda (test1 test2 projekt1 projekt2)
    (let ((m (if (< (min test1 test2 projekt1 projekt2) 0)
                 (min test1 test2 projekt1 projekt2) ; Täuschung
                 (max test1 test2 projekt1 projekt2)))) ; Keine Täuschung
      (let ((p (if (< m 0) 0 m)))
        (if (= m test1)
            (+ p (tests-projekte-punkte-1 test2 projekt1 projekt2))
            (if (= m test2)
                (+ p (tests-projekte-punkte-1 test1 projekt1 projekt2))
                (if (= m projekt1)
                    (+ p (tests-projekte-punkte-1 test1 test2 projekt2))
                    (+ p (tests-projekte-punkte-1 test1 test2 projekt1))))))))))

;;; SIGNATUR
;;; tests-projekte-punkte-1: number number number -> number
;;; ERKLÄRUNG
;;; (tests-projekte-punkte-1 p1 p2 p3) liefert von drei Punktzahlen aus Tests
;;; und Programmierprojekten diejenige zurück, die zur Berechnung der Note
;;; verwendet wird.
;;; BEISPIEL
;;; (tests-projekte-punkte-1 0 5 13)
;;; => 13
;;; DEFINITION
(define tests-projekte-punkte-1
  (lambda (p1 p2 p3)
    (if (< (min p1 p2 p3) 0)
        0 ; Täuschung
        (max p1 p2 p3))) ; Keine Täuschung

```

