

---

PROBEKLAUSUR ZU  
'SOFTWAREENTWICKLUNG 3 (LOGIKPROGRAMMIERUNG)'

WOLFGANG MENZEL

---

**Die Klausur bitte auf jeden Fall geheftet lassen! Kein zusätzliches Papier verwenden!**

**Name, Vorname:** \_\_\_\_\_

**Matrikelnummer:** \_\_\_\_\_

**Studiengang:** \_\_\_\_\_

**Unterschrift:** \_\_\_\_\_

**WICHTIG:** Punkte für Teilaufgaben werden durch  $\{\circ\}$  angegeben, z.B.  $\{\circ \circ\} = 2$  Punkte. Die Gesamtanzahl an Punkten pro Aufgabe steht jeweils im Kasten am Rand der betreffenden Aufgabe unter "von".

Der vorgesehene Platz für eine Aufgabe entspricht etwa dem Umfang der von Ihnen erwarteten Lösung. Sollte der Platz dennoch nicht ausreichen, so verwenden Sie bitte die Leerseiten am Ende des Klausurbogens und machen einen Vermerk am Rand, etwa " $\Rightarrow$  Seite 15".

1. Eine relationale Datenbank für Bibliotheksausleihen sei als Faktensammlung in Prolog implementiert und enthält die Relationen `ausleihe(Signatur, Lesernummer, Ausleihdatum)` für jedes derzeit ausgeliehene Buch, `vorbestellung(Signatur, Lesernummer)` für jedes derzeit vorbestellte Buch, und `leser(Name, Vorname, Lesernummer, Adresse, Geburtsjahr)` für die persönlichen Angaben jedes Lesers. Das Ausleihdatum ist als dreistellige Struktur `datum(Jahr, Monat, Tag)` implementiert. Stellen Sie die folgenden Anfragen an die Datenbank. Formulieren Sie ihre Anfragen so, dass möglichst wenig überflüssige Information präsentiert wird und die Systemausgabe weitgehend selbsterklärend ist.

von
10

- (a) Welche Lesernummer hat Susi Sorglos?  $\{\circ\}$
- (b) Welcher Leser (identifiziert durch Name und Vorname) hat das Buch mit der Signatur BUG17456 vorbestellt?  $\{\circ\circ\}$
- (c) Welche Bücher, die der Leser mit der Lesernummer 56245 ausgeliehen hat, können verlängert werden (d.h. sind nicht vorbestellt)? Hinweis: Die Negation einer Bedingung kann durch das Prädikat `\+/1` erfolgen.  $\{\circ\circ\}$

- (d) Ermitteln Sie den Vornamen, den Namen, die Adresse und die Signatur des ausgeliehenen Buches für alle Leser, die vor dem 7.1.2007 ein Buch ausgeliehen haben. { } { }

- (e) Wieviele Leser, die älter als 60 Jahre sind, haben ein derzeit mindestens ein Buch ausgeliehen? { } { }

2. Für die Arbeit mit der Bibliotheks-Datenbank aus Aufgabe 2 werden eine Reihe zusätzlicher Prädikate benötigt.

(a) Definieren Sie ein Ordnungsprädikat für zwei Datumsangaben.  $\{\emptyset\}$

<b>von</b>
<b>12</b>

(b) Definieren Sie ein Prädikat, dass die Differenz (in Tagen) zwischen zwei Datumsangaben ermittelt. Schaltjahre sollen dabei unberücksichtigt bleiben.  $\{\emptyset \emptyset \emptyset \emptyset \emptyset\}$

- (d) Zur bequemen Ermittlung von Leihfristüberschreitungen, wird ein Prädikat gewünscht, das für das aktuelle Datum und eine als Parameter einzugebende Leihfrist (in Tagen) die Nummern der Leser ermittelt, die die Leihfrist überzogen haben. Formulieren Sie das Prädikat so, dass die Resultate als alternative Variablenbindungen ausgegeben werden.
- $\{\emptyset \emptyset\}$

- (e) Um beim Versand der Mahnungen Postgebühren zu sparen, sollen die Informationen für einen Leser in jeweils einem Anschreiben zusammengefasst werden. Definieren Sie ein Prädikat, das für eine gegebene Lesernummer, das aktuelle Datum und die maximale Leihfrist, die Zahl der zurückzugebenden Bücher und eine *Liste* der zugehörigen Signaturen ermittelt.
- $\{\emptyset \emptyset \emptyset \emptyset\}$

3. Gegeben sei erneut die Bibliotheks-Datenbank aus Aufgabe 2.

- (a) Definieren Sie ein Prädikat, das eine *Liste* ermittelt, in der alle zu mahnenden Leser mit Angaben zu Name, Vorname und Adresse, sowie zur Anzahl der zurückzugebenden Bücher und deren Signaturen enthalten sind.

Hinweis: Um Duplikate zu entfernen, können Sie die Liste mit `sort/2` sortieren, oder bereits duplikatenfrei mit `setof/3` erzeugen.

$\{\emptyset \emptyset \emptyset \emptyset \emptyset \emptyset\}$

von
10

- (b) Die Bibliothek möchte die Alterszusammensetzung ihrer aktiven Leser ermitteln. Definieren Sie ein Prädikat, das die Anzahl der Leser in vorgegebenen Altersgruppen bestimmt. Die Altersgruppen sind durch eine Liste von Intervallgrenzen definiert, z.B.:

$[0, 10, 14, 18, 28, 38, 50, 65, 500]$

In der Statistik sollen nur diejenigen Leser berücksichtigt werden, die derzeit mindestens ein Buch ausgeliehen haben.

Hinweis: Definieren Sie sich ein Hilfsprädikat, das die Anzahl der Leser für *ein* gegebenes Intervall ermittelt, und arbeiten Sie dann die Liste der Intervallgrenzen rekursiv ab.  $\{\circ \circ \circ \circ\}$

4. Gegeben sind die folgenden drei Prädikatsdefinitionen:

$p1([ ], [ ]).$   
 $p1([A, \_ | X], [A, \_ | Y]) :- p1(X, Y).$

$p2([ ], [ ]).$   
 $p2([A], [A]).$   
 $p2([A, \_ | X], [A, \_ | Y]) :- p2(X, Y).$

$p3([ ], [ ]).$   
 $p3([ ], [ ]).$   
 $p3([A, \_ | X], [A, \_ | Y]) :- p3(X, Y).$

- (a) Was berechnen die drei Prädikate  $p1/2$ ,  $p2/2$  und  $p3/2$ ? Was unterscheidet sie?  $\{\circ \circ\}$

- (b) Ersetzen Sie die in den Prädikatsdefinitionen verwendeten Namen durch selbsterklärende Bezeichner. Vervollständigen Sie die Definitionen durch geeignete Kommentare!  $\{\circ \circ\}$

von
10



- (c) Mit welchen Aufrufen würden Sie das ursprüngliche Prädikat  $p1/2$  testen? Welche Resultate erwarten Sie jeweils?  $\{\circ \circ \circ\}$

- (d) Welche Eigenschaften (symmetrisch, reflexiv, transitiv, eindeutig im ersten Argument (1:m), eindeutig im zweiten Argument (m:1)) haben die durch  $p2/2$  und  $p3/2$  definierten Relationen?  $\{\circ \circ \circ\}$

Zutreffendes ankreuzen		symmetrisch	reflexiv	transitiv	1:m	m:1
$p2/2$	Ja	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	Nein	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
$p3/2$	Ja	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
	Nein	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

5. (a) Wozu dient die funktionale Auswertungsumgebung in Prolog, warum braucht man sie und was ist bei Ihrer Verwendung zu beachten?  
{ $\emptyset$   $\emptyset$ }

von
6

- (b) Welche Systemreaktionen erwarten Sie bei den folgenden Eingaben am Systemprompt? Warum? { $\emptyset$   $\emptyset$   $\emptyset$   $\emptyset$ }

- ?- X is 3 + 4 \* 2.

- ?- X = 3 + 4 \* 2.

- ?- X < 3 + 4 \* 2.

- ?- X is 3, Y is 4, X + 1 is Y.

## 6. Funktionale Programmierung

(a) Wozu evaluieren die folgenden Scheme-Ausdrücke?

i. `(cons (cdr (quote ((1 2) 3)))  
         (car (quote ((1 2) 3))))`  $\{\emptyset\}$

ii. `(map (lambda (x) (+ 1 x))  
         (quote (1 2 3)))`  $\{\emptyset\}$

iii. `(filter (curry > 0)  
         (quote (2 -3 1 4 -1 3 -2 0)))`  $\{\emptyset\}$

(b) Wozu evaluieren die folgenden Ausdrücke, wenn sie in dieser Reihenfolge am Systemprompt eines Scheme-Systems eingegeben werden?  
 $\{\emptyset \emptyset\}$

```
(define x 4)
(define y 3)
(define (foo x)
  (* (+ 4 y) x) )
(foo x)
(foo y)
```

von
12

- (c) Was sind special form expressions in Scheme und warum braucht man sie?  $\{\emptyset\}$

- (d) Erklären Sie, was die folgende Funktion berechnet.  $\{\emptyset\}$

```
;; a ist eine Liste, n eine natuerliche Zahl
(define (nimm a n)
  (if (null? a)
      '()
      (if (eq? n 0)
          '()
          (cons (car a) (nimm (cdr a) (- n 1)))
      )
  )
)
```

(e) Geben sie für die Funktion aus Aufgabenteil (d) eine entsprechende Prädikatsdefinition in Prolog an.  $\{\circ\circ\}$

(f) Diskutieren Sie eventuelle Unterschiede im Verhalten der Programme aus Aufgabenteil (d) und (e).  $\{\circ\}$

## Wichtige Prolog-Prädikate

<code>last(?Liste,?Element)</code>	Element ist das letzte Element von Liste
<code>length(?Liste,?Länge)</code>	Länge ist die Anzahl der Elemente in Liste
<code>nth1(?Index,?Liste,?Element)</code>	Element steht an der Position Index in Liste
<code>nth0(?Index,?Liste,?Element)</code>	wie <code>nth1/2</code> aber erstes Element hat Index 0
<code>member(?Element,?Liste)</code>	Element ist in Liste enthalten
<code>memberchk(?Element,?Liste)</code>	Äquivalent zu <code>member/2</code> aber hinterlässt keine Entscheidungspunkte
<code>append(?Liste1,?Liste2,?Liste3)</code>	Verkettet Liste1 mit Liste2 zu Liste3
<code>sumlist(+Liste,?Summe)</code>	Ermittelt für eine Liste von Zahlen deren Summe
<code>sublist(+Praedikat,+Liste1,?Liste2)</code>	Sammelt alle Elemente aus Liste1, für die das einstellige Praedikat wahr ist in Liste2
<code>findall(+Muster,+Ziel,?Liste)</code>	Liefert eine Liste von Elementen des Typs Muster, die durch den Aufruf von Ziel instanziiert wurden
<code>setof(+Muster,+Ziel,?Liste)</code>	wie <code>findall/3</code> allerdings ist die Ergebnisliste sortiert und duplikatenfrei
<code>sort(+Liste1,?Liste2)</code>	Sortiert Liste1 und beseitigt Duplikate
<code>msort(+Liste1,?Liste2)</code>	wie <code>sort/2</code> , allerdings bleiben Duplikate erhalten

## Wichtige Scheme-Funktionen

<code>(zero? x)</code>	Testet, ob die Zahl $x$ gleich null ist.
<code>(positive? x)</code>	Testet, ob die Zahl $x$ positiv ist. Die Null ist nicht positiv.
<code>(negative? x)</code>	Testet, ob die Zahl $x$ negativ ist. Die Null ist nicht negativ.
<code>(&lt; x y)</code>	Testet, ob die Zahl $x$ kleiner als die Zahl $y$ ist.
<code>(&gt; x y)</code>	Testet, ob die Zahl $x$ größer als die Zahl $y$ ist.
<code>(+ x1 ... xn)</code>	Addiert die Zahlen $x1$ bis $xn$ .
<code>(- x)</code>	Wechselt das Vorzeichen einer Zahl $x$ .
<code>(- x1 x2)</code>	Subtrahiert die Zahl $x2$ von der Zahl $x1$ .
<code>(* x1 ... xn)</code>	Multipliziert die Zahlen $x1$ bis $xn$ .
<code>(expt x y)</code>	Potenziiert $x$ mit dem Exponenten $y$ .
<code>(abs x)</code>	Ermittelt den Absolutbetrag von $x$ .
<code>(null? x)</code>	Testet, ob die Liste $x$ leer ist.
<code>(car x)</code>	Ermittelt das erste Element einer Liste
<code>(cdr x)</code>	Ermittelt den Rest einer Liste nach Entfernen des ersten Elements.
<code>(cons x1 x2)</code>	Baut eine Liste, indem es das Element $x1$ an den Anfang einer Liste $x2$ setzt.
<code>(list x1 ... xn)</code>	Fügt die Elemente $x1$ bis $xn$ zu einer Liste zusammen.
<code>(member? x1 x2)</code>	Testet, ob das Element $x1$ in Liste $x2$ enthalten ist.
<code>(map f x1 ... xn)</code>	Wendet die ( $n$ -stellige) Funktion $f$ auf die Listenelemente von $x1$ ... $xn$ an und fasst die Auswertungsergebnisse in einer Liste zusammen
<code>(filter f x)</code>	Wendet die Boole'sche Funktion $f$ auf alle Elemente der Liste $x$ an und übernimmt alle diejenigen Listenelemente in das Ergebnis, für die die Auswertung von $f$ #t ergibt.
<code>(curry f x)</code>	Bindet das erste Argument einer zweistelligen Funktion $f$ an den Wert von $x$ .
<code>(rcurry f x)</code>	Bindet das zweite Argument einer zweistelligen Funktion $f$ an den Wert von $x$ .
<code>(compose f1 f2)</code>	Erzeugt eine neue Funktion, die die Funktionen $f2$ und $f1$ nacheinander auswertet.

Aufgabe	Punkte
1	
2	
3	
4	
5	
6	
Summe:	

	<b>Bestanden?</b>	
von	Ja <input type="radio"/>	Nein <input type="radio"/>
60		

Bestanden bei $\geq 30$ Punkten		
Punkte	Note	Bestanden?
57 – 60	1	Ja
54 – 58.5	1-	Ja
51 – 53.5	2+	Ja
48 – 50.5	2	Ja
45 – 47.5	2-	Ja
42 – 44.5	3+	Ja
39 – 41.5	3	Ja
36 – 38.5	3-	Ja
33 – 35.5	4+	Ja
30 – 32.5	4	Ja
0 – 29.5	5	Nein

**Datum & Unterschrift Vorkorrektur** \_\_\_\_\_

**Datum & Unterschrift PrüferIn 1** \_\_\_\_\_

**Datum & Unterschrift PrüferIn 2** \_\_\_\_\_