# PROBEKLAUSUR ZU 'SOFTWAREENTWICKLUNG 3 (LOGIKPROGRAMMIERUNG)'

#### WOLFGANG MENZEL

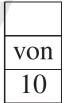
## Die Klausur bitte auf jeden Fall geheftet lassen! Kein zusätzliches Papier verwenden!

Name, Vorname:	
Matrikelnummer:	
Studiengang:	
Unterschrift:	

**WICHTIG:** Punkte für Teilaufgaben werden durch  $\{\emptyset\}$  angegeben, z.B.  $\{\emptyset \emptyset\} = 2$  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 "⇒ 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.



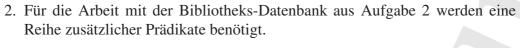
(a) Welche Lesernummer hat Susi Sorglos?

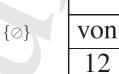
 $\{\oslash\}$ 

(b) Welcher Leser (identifiziert durch Name und Vorname) hat das Buch mit der Signatur BUG17456 vorbestellt? {⊘⊘}

(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. {∅ ∅} (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?  $\{ \oslash \oslash \bigcirc \}$ 





(a) Definieren Sie ein Ordnungsprädikat für zwei Datumsangaben.

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

(d) Zur bequemeren 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. {∅ ∅}

(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.

 $\{ \oslash \oslash \oslash \oslash \}$ 

- 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.

von 10

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

 $\{ \oslash \oslash \oslash \oslash \oslash \oslash \}$ 

(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.:

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.  $\{ \oslash \oslash \oslash \bigcirc \}$ 

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?{∅ ∅}

von

10

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



(c) Mit welchen Aufrufen würden Sie das ursprüngliche Prädikat p1/2 testen? Welche Resultate erwarten Sie jeweils? {∅ ⊘ ⊘}

(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?  $\{ \oslash \oslash \oslash \}$ 

Zutreff	endes ankreuzen	symmetrisch	reflexiv	transitiv	1:m	m:1
p2/2	Ja	0	0	0	0	0
	Nein		$\circ$	0	$\bigcirc$	$\bigcirc$
p3/2	Ja		$\bigcirc$	0		
	Nein	Ó				

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





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

• 
$$?- X is 3 + 4 * 2.$$

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

• 
$$?- X < 3 + 4 * 2.$$

#### 6. Funktionale Programmierung

(a) Wozu evaluieren die folgenden Scheme-Ausdrücke?

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

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

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

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

```
inangio den du.
```

(d) Erklären Sie, was die folgende Funktion berechnet.

 $\{\oslash\oslash\}$ 

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

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

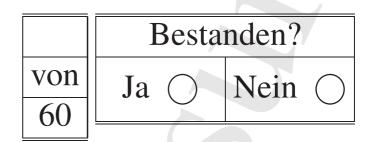
### Wichtige Prolog-Prädikate

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

### Wichtige Scheme-Funktionen

(zero? x)	Testet, ob die Zahl x gleich null ist.
(positive? x)	Testet, ob die Zahl x positiv ist. Die Null ist nicht positiv.
(negative? x)	Testet, ob die Zahl x negativ ist. Die Null ist nicht negativ.
(< x y)	Testet, ob die Zahl x kleiner als die Zahl y ist.
(> x y)	Testet, ob die Zahl x größer als die Zahl y ist.
(+ x1 xn)	Addiert die Zahlen x1 bis xn.
(- X)	Wechselt das Vorzeichen einer Zahl x.
(- x1 x2)	Subtrahiert die Zahl x2 von der Zahl x1.
(* x1 xn)	Multipliziert die Zahlen x1 bis xn.
(expt x y)	Potenziert x mit dem Exponenten y.
(abs x)	Ermittelt den Absolutbetrag von x
(null? x)	Testet, ob die Liste x leer ist.
(car x)	Ermittelt das erste Element einer Liste
(cdr x)	Ermittelt den Rest einer Liste nach Entfernen des ersten Elements.
(cons x1 x2)	Baut eine Liste, indem es das Element x1 an den Anfang einer Liste x2 setzt.
(list $x1 \dots xn$ )	Fügt die Elemente x1 bis xn zu einer Liste zusammen.
(member? x1 x2)	Testet, ob das Element x1 in Liste x2 enthalten ist.
(map f x1 xn)	Wendet die (n-stellige) Funktion $f$ auf die Listenelemente von $x1 \dots xn$ an und fasst die Auswertungsergebnisse in einer Liste zusammen
(filter f x)	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.
(curry f x)	Bindet das erste Argument einer zweistelligen Funktion f an den Wert von x.
(rcurry f x)	Bindet das zweite Argument einer zweistelligen Funktion f an den Wert von x.
(compose f1 f2)	Erzeugt eine neue Funktion, die die Funktionen f2 und f1 nacheinander auswertet.

Aufgabe	Punkte
1	
2	
3	
4	
5	
6	
Summe:	



Bestanden bei		
≥ 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 PriiferIn 2	