PROBEKLAUSUR ZU
' SOFTWAREENTWICKLUNG 3 (LOGIKPROGRAMMIERUNG)'
WS 2012/2013

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 $\{\oslash\}$ angegeben, z.B. $\{\oslash\ \oslash\}$ = 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 "$\Longrightarrow$ Seite 15".

von
10

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? {⊘}

   Lösung:

   ```
   ?- leser('Sorglos','Susi',Lesernummer,_,_).
   ```

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

   ```
   ?- leser(Name,Vorname,Lesernummer,_,_),
      vorbestellung('BUG17456',Lesernummer).
   ```

   (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. {⊘ ⊘}

   Lösung:

   ```
   ?- ausleihe(Signatur,562456),
      \+ vorbestellung(Signatur,_).
   ```

(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. {⊘ ⊘}

Lösung:

```
?- ausleihe(Signatur,Lesernummer,datum(Jahr,Monat,Tag)),
   (Jahr<2007 ;
    Jahr=2007,Monat=1,Tag<7),
   leser(Name,Vorname,Lesernummer,Adresse,_).
```

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

Lösung:

```
?- setof(Lesernr,(ausleihe(_,Lesernr,_),leser(_,_,Lesernr,_,Jahr),J
   length(Leser,Anzahl).
```

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

von
12

(a) Definieren Sie ein Ordnungsprädikat für zwei Datumsangaben. {⊘}

Lösung:

```
vor(D1,D2) :- D1@<D2.
```

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

Lösung:

```
diff(datum(Y,M,T1),datum(Y,M,T2),Diff) :- Diff is T2 - T1.
diff(datum(J,M1,T1),datum(J,M2,T2),Diff) :-
   M2>M1,
   member(M1,[1,3,5,7,8,10,12]),
   Mx is M1 + 1,
   diff(datum(J,Mx,T1),datum(J,M2,T2),D),
   Diff is D + 31.
diff(datum(J,2,T1),datum(J,M2,T2),Diff) :-
   M2>2,
   diff(datum(J,3,T1),datum(J,M2,T2),D),
   Diff is D + 28.
diff(datum(J,M1,T1),datum(J,M2,T2),Diff) :-
   M2>M1,
   member(M1,[4,6,9,11]),
   Mx is M1 + 1,
   diff(datum(J,Mx,T1),datum(J,M2,T2),D),
   Diff is D + 30.
diff(datum(J1,M1,T1),datum(J2,M2,T2),Diff) :-
   J2>J1,
   Jx is J1 + 1,
   diff(datum(Jx,M1,T1),datum(J2,M2,T2),D),
   Diff is D + 365.
```

(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.
{⊘ ⊘}

Lösung:

```
im_verzug(Aktuelles_Datum,Leihfrist,Lesernummer) :-
   ausleihe(Signatur,Lesernummer,Datum_Ausleihe),
   diff(Datum_Ausleihe,Aktuelles_Datum,Diff),
   Diff > Leihfrist.
```

(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.
{⊘ ⊘ ⊘ ⊘}

Lösung:

```
muss_zurückgeben(Lesernummer,Aktuelles_Datum,Leih-
   frist,Anzahl,Liste_B findall(Signatur,
            (ausleihe(Signa-
              tur,Lesernummer,Datum_Ausleihe
              ),
              diff(Aktuelles_Datum,Datum_Aus
              leihe,Diff),
              Diff > Leihfrist),
            Liste_Buecher),
   length(Liste_Buecher,Anzahl).
```

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

von
10

(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.
{⊘ ⊘ ⊘ ⊘ ⊘ ⊘}

Lösung:

```
mahnungsliste(Datum,Liste) :-
   setof(Lesernummer,im_verzug(Da-
   tum,Lesernummer),Liste_LN),
   findall([Name,Vorname,Adresse,Anzahl,Signatu-
       ren], forall(member(LN,Liste_LN),
            (leser(Name,Vorname,Adresse,Lesernummer,_),
              muss_zurückgeben(LN,Datum,Anzahl,Signatu-
              ren))).
```

(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. {⊘ ⊘ ⊘ ⊘}

Lösung:

```
in_quantil(U,O,Anzahl) :-
  setof(Lesernummer,
        (ausleihe(_,_,_,Lesernummer,Alter>U,Alter=<O)),
        Liste),
  length(Liste,Anzahl).

% statistik(Quantile,Distribution) :-
statistik([_],[]).
statistik([U,O|R],[Anzahl|R1]) :-
   in_quantil(U,O,Anzahl).
```

4. Gegeben sind die folgenden drei Prädikatsdefinitionen:

von
10

```
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? {⊘ ⊘}

(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?
{⊘ ⊘ ⊘}

Zutreffendes ankreuzen		symmetrisch	reflexiv	transitiv	1:m	m:1
`p2/2`	Ja	◯	◯	◯	◯	◯
	Nein	◯	◯	◯	◯	◯
`p3/2`	Ja	◯	◯	◯	◯	◯
	Nein	◯	◯	◯	◯	◯

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

von
6

Lösung:

- Zur Auswertung arithmetischer Ausdrücke.
- Weil sie effiziente numerische Berechnungen ermöglicht.
- Sie ist richtungsabhängig, d.h. alle Variablen in dem auszuwertenden Ausdruck müssen vollständig instanziiert sein.

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

- `?- X is 3 + 4 * 2.`
  Lösung:

```
X = 11 ;
No
```

- `?- X = 3 + 4 * 2.`
  Lösung:

```
X = 3 + 4 * 2 ;
No
```

- `?- X < 3 + 4 * 2.`
  Lösung:

```
Error: Arguments are not sufficiently instantiated
```

- `?- X is 3, Y is 4, X + 1 is Y.`
  Lösung:

```
No
```

6. Funktionale Programmierung

von
12

(a) Wozu evaluieren die folgenden Scheme-Ausdrücke?

i.
```
(cons (cdr (quote ((1 2) 3)))
      (car (quote ((1 2) 3))))
```
{⊘}

Lösung: ((3) 1 2)

ii.
```
(map (lambda (x) (+ 1 x))
     (quote (1 2 3)))
```
{⊘}

Lösung: (2 3 4)

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

Lösung: (-3 -1 -2)

(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)
```

Lösung:

```
==> undefined
==> undefined
==> undefined
==> 28
==> 21
```

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

Lösung: Special form expressions sind Ausdrücke, deren Auswertungsverhalten von der Standard-Auswertungsreihenfolge abweicht. Z.B. können Argumente gar nicht, teilweise oder bedingt ausgewertet werden. Special form expressions werden z.B. für Steuerstrukturen, z.B. (if <cond> <then> <else>) benötigt, wo in Abhängigkeit vom Wahrheitswert der Bedingung nur bestimmte Argumente (entweder das zweite oder das dritte Argument) ausgewertet werden dürfen.

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

```
;; 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)))
  ) ) )
```

Lösung: `nimm` ermittelt die Liste, die die ersten `n` Elemente der Liste `a` enthält (Präfix der Länge `n`). Falls die Länge der Liste `a` kürzer als `n` Elemente ist, gibt die Funktion die Liste `a` unverändert zurück.

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

Lösung:

```
% nimm(Liste,Laenge,Praefix)
nimm([],_,[]).
nimm(_,0,[]).
nimm([H|R],N,[H|R1]) :- N1 is N-1, nimm(R,N1,R1).
```

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

Lösung: Das Prolog-Prädikat kann auch in weiteren Instanziierungsvarianten aufgerufen werden (z.B. als Test). Das zweite Argument muss aber immer instanziiert werden.

Aufgabe	Punkte
1	
2	
3	
4	
5	
6	
Summe:	

	Bestanden?	
von	Ja ◯	Nein ◯
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 _____