Упражнение 1 по Пролог (LPA Win-Prolog)

(1.11.2004 г.)

I. Основни понятия.

Програма на Пролог – съвкупност от **клаузи** на Хорн.

Клаузите имат вида:

a :- b_1, …, b_j. или

a if b_1 and … and b_j.

Ако няма нищо отдясно на :- (на if), се записва

a.

и се нарича **факт**, иначе – **правило**.

Правилата и фактите описват отношения в предметната област (ПО) и свойствата на обектите. Фактите задават безусловно верни твърдения.

Например, фактът

male(ivan).

задава, че Иван е от мъжки пол.

ivan е **константа** и съответства на конкретен обект от ПО.

male е едноаргументен **предикат**, който представя свойство на обекти от ПО.

Символните константи започват с малка буква. Има още числови (12, -3.456, 2.21e-3) и стрингови (използват се кавички ("Паисий Хилендарски")) константи.

Пролог приема за вярно само това, което е зададено явно. (Ако няма факт, че Петър е мъж, то Пролог приема, че той не такъв.)

Правилото съответства на импликация $\mathbf{b_1 \wedge \ldots \wedge b_j \rightarrow a}$ и може да бъде прочетено – **a** е вярно, ако са верни $\mathbf{b_1}$ и … и $\mathbf{b_j}$.

Например, правилото

likes(ivan,X):-male(Y),likes(Y,X).

задава, че Иван харесва всички неща, които се харесват от някой (кой да е) мъж.

X и Y са **променливи** (започват с *голяма* буква). likes е двуаргументен **предикат** и представя отношение между обекти.

Програмата може да съдържа и клауза без лява част, т.е.

:- b_1,…,b_j.

или

?- **b_1,…,b_j.**

Такава клауза се нарича **цел** или **въпрос**. Целта представя задачата, която искаме да реши Пролог – т.е. да провери дали е вярно $\mathbf{b_1, \ldots, b_j}$.

Всъщност програмата се разглежда като съвкупност от аксиоми, а въпроса – като логическо следствие (теорема), което трябва да бъде изведено от тези аксиоми. Доказателството на целта се извършва по метода на резолюцията (селективна линейна резолюция) – към множеството от аксиоми се добавя отрицанието на целта, която ще се доказва (записва се ?- b_1,…,b_j.) и трябва да се изведе празната клауза (:-), т.е. да се докаже, че полученото след добавянето на отрицанието на целта множество от формули е неудовлвтворимо (противоречиво).

При извода се използва правилото на резолюцията, което в нотацията на Пролог може да бъде формулирано така:

от клаузите p:- $q_1, \ldots, q_{i-1}, \boldsymbol{q_i}, q_{i+1}, \ldots, q_n$ и $\boldsymbol{q_i}$:- $r_1, \ldots, r_k$

се извежда клаузата p:- $q_1, \ldots, q_{i-1}, r_1, \ldots, r_k, q_{i+1}, \ldots, q_n$.

Ако има променливи търсим подходяща субституция (заместване) на променливите с терми (константи, променливи или съставни терми), така че да се получат еднакви терми, в лявата страна на едната клауза и отдясно в другата.

Задача: С помощта на метода на резолюцията проверете дали от клаузите

1. s:- .
2. t:- .
3. p:- v.
4. p:- r,t.
5. u:- p,q.
6. q:- t.
7. v:- q.
8. r:- s.

може да се изведе клаузата u.

Решение (Заб.: Има 2 пътя на доказателство в зависимост от това коя клауза за p използваме.):
Добавяме отрицанието на целта.

9. :- u.

От 9 и 5 получаваме 10. :- p,q.
От 10 и 3 получаваме 11. :- v,q.
От 11 и 7 получаваме 12. :- q,q.
От 12 и 6 получаваме 13. :- t,q.
От 13 и 2 получаваме 14. :- q.
От 14 и 6 получаваме 15. :- t.
От 15 и 6 получаваме :- . (празната клауза), от което следва, че u е вярно (u е следствие от даденото множество клаузи).

Задача: Чрез метода на резолюцията изведете t(a,b) от клаузите:

1. s(a):-.
2. s(b):-.
3. p(X):-q(X),r(X).
4. t(X,Y):-s(X),p(Y).
5. q(X):-u(X).
6. r(X):-u(X).
7. u(X):-s(X).

Решение:
Добавяме

8. :-t(a,b).

От 8 и 4, и субституция {a/X,b/Y} получаваме

9. :-s(a),p(b).

От 9 и 1 получаваме

10. :-p(b).

От 10 и 3, и субституция {b/X} получаваме

11. :-q(b),r(b).

От 11 и 5, и субституция {b/X} получаваме

12. :-u(b),r(b).

От 12 и 7, и субституция {b/X} получаваме

13. :-s(b),r(b).

От 13 и 2 получаваме

14. :-r(b).

От 14 и 6, и субституция {b/X} получаваме

15. :-u(b).

От 15 и 7, и субституция {b/X} получаваме

16. :-s(b).

От 16 и 2 получаваме

17. :- .

След добавяне на отрицанието на целта изведохме празната клауза, от което следва, че целта е изводима от даденото мн-во клаузи.

Задача: Чрез метода на резолюцията докажете, че множеството клаузи {a:-.; b:-.; c:-.; s:-d,f.; s:-d,b,v.; f:-a,c.; d:-b.; :-s.} е противоречиво.

Задачи:

Зад. 1. Дадени са факти на Пролог за това коя страна на кой континент се намира, коя е столицата на дадена страна и за населението на някои градове. Задайте следните цели:

1. България / Магьосия в Европа ли е?
2. В кой континент е България / Русия?
3. Кои са страните в Европа?
4. Кои са европейските столици (столиците на страни в Европа)?
5. Кои са градовете с население над 3 млн.?
6. Кои са столиците с население над 7 млн?
7. Кои са столиците в Европа с население над 7 млн.? (Да се извежда само столицата)
8. Има ли два града с еднакъв брой жители?
9. Кои са континентите, в които има страни със столица с над 4 млн жители?
10. Кои са градовете, които не са столица на някоя страна?
11. Кои са градовете, чието население е по-малко от населението на някой друг град?
12. Като 11, но да се извеждат еднократно градовете.
13. Кой град (градове) има най-малко население?
14. Кои са страните в Азия или Евроопа?

```
 /* Програма continent */
%  str_kont(S,K) означава, че страната с име S е в континента с име K
str_kont('Египет','Африка').
str_kont('Русия','Европа').
str_kont('Русия','Азия').
str_kont('Франция','Европа').
str_kont('Япония','Азия').
str_kont('Магьосия','Европа').
 %  stolica_str(L,S) означава, че  градът с име L е столица на страната с име S
stolica_str('Талин','Естония').
stolica_str('Париж','Франция').
stolica_str('Токио','Япония').
stolica_str('Москва','Русия').
stolica_str('Абра Кадабра','Магьосия').
 %  gr_nasel(G,N) означава, че  градът с име G има население с брой N
gr_nasel('София',1850000).
gr_nasel('Токио',7500500).
gr_nasel('Москва',21000300).
gr_nasel('Париж',6600500).
gr_nasel('Берлин',9000888).
gr_nasel('Пловдив',550000).
gr_nasel('Първомай',95000).
gr_nasel('Силистра',95000).
gr_nasel('Карнобат',95000).
gr_nasel('Абра Кадабра',33333333).

/*---------- Допълнителни предикати, използвани като помощни в целите ------*/
bigEuropeStol(Stol):-
    str_kont(Str,'Европа'),stolica_str(Stol,Str),gr_nasel(Stol,N),N>7000000.
ima:- gr_nasel(X,Y),gr_nasel(Z,Y),X\=Z.
kont(X):- str_kont(Y,X),stolica_str(Z,Y),gr_nasel(Z,N),N>4000000.
min(X):- gr_nasel(X,N),\+ (gr_nasel(_,N1),N>N1).
p(G):-gr_nasel(G,N),gr_nasel(_,N1),N<N1.
p1(Gr):- gr_nasel(Gr,N),one (gr_nasel(_,N1),N<N1).

/* Въпроси (цели):
1. България в Европа ли е?
     ?- str_kont('България','Европа').
     Отговор: no
     (Няма факт България в кой континент е.
      Ако не е указано явно, Пролог го приема за невярно.)
```

```
    Магьосия в Европа ли е?
    ?- str_kont('Магьосия','Европа').
    Отговор: yes
 2. В кой континент е България?
    ?- str_kont('България',X).
    Отговор: no
    В кой континент е Русия?
    ?- str_kont('Русия',X).
    Отговор: X = 'Европа' ;
             X = 'Азия'
 3. Кои са страните в Европа?
    ?- str_kont(X,'Европа').
    Отговор: X = 'Русия' ;
             X = 'Франция' ;
             X = 'Магьосия'
4. Кои са европейските столици (столиците на страни в Европа)?
    ?- stolica_str(Stolica,Y),str_kont(Y,'Европа').
    Отговор: Stolica = 'Париж' ,
             Y = 'Франция' ;
             Stolica = 'Москва' ,
             Y = 'Русия' ;
             Stolica = 'Абра Кадабра' ,
             Y = 'Магьосия'
5. Кои са градовете с население над 3 млн.?
    ?- gr_nasel(Grad,N),N>3000000.
    Отговор: Grad = 'Токио' ,
             N = 7500500 ;
             Grad = 'Москва' ,
             N = 21000300 ;
             Grad = 'Париж' ,
             N = 6600500 ;
             Grad = 'Берлин' ,
             N = 9000888 ;
             Grad = 'Абра Кадабра' ,
             N = 33333333
6. Кои са столиците с население над 7 млн?
    ?- stolica_str(Stolica,_),gr_nasel(Stolica,Nasel),Nasel>7000000.
    Отговор: Stolica = 'Токио' ,
             Nasel = 7500500 ;
             Stolica = 'Москва' ,
             Nasel = 21000300 ;
             Stolica = 'Абра Кадабра' ,
             Nasel = 33333333
7. Кои са столиците в Европа с население над 7 млн.? (Да се извежда само столицата)
    Помощен предикат:
    bigEuropeStol(Stol):-
       str_kont(Str,'Европа'),stolica_str(Stol,Str),gr_nasel(Stol,N),N>7000000.
    Въпрос:
    ?- bigEuropeStol(X).
    Отговор: X = 'Москва' ;
             X = 'Абра Кадабра'
8. Има ли два града с еднакъв брой жители?
    ?- gr_nasel(X,Y),gr_nasel(Z,Y),X\=Z.
    Отговор: X = 'Първомай' ,
             Y = 95000 ,
             Z = 'Силистра' ;
             X = 'Първомай' ,
             Y = 95000 ,
             Z = 'Карнобат' ;
             X = 'Силистра' ,
             Y = 95000 ,
             Z = 'Първомай' ;
```

```
            X = 'Силистра' ,
            Y = 95000 ,
            Z = 'Карнобат' ;
            X = 'Карнобат' ,
            Y = 95000 ,
            Z = 'Първомай' ;
            X = 'Карнобат' ,
            Y = 95000 ,
            Z = 'Силистра' ;
            no
```

Или с допълнителен предикат: `ima:- gr_nasel(X,Y),gr_nasel(Z,Y),X\=Z.`

```
?- ima.
Отговор: yes
```

9. Кои са континентите, в които има страни със столица с над 4 млн жители?

Допълнителен предикат:

```
kont(X):- str_kont(Y,X),stolica_str(Z,Y),gr_nasel(Z,N),N>4000000.
?- kont(X).
Отговор: X = 'Европа' ;
         X = 'Азия' ;
         X = 'Европа' ;
         X = 'Азия' ;
         X = 'Европа'
```

10. Кои са градовете, които не са столица на някоя страна?

```
?- gr_nasel(G,_),\+ stolica_str(G,_).
Отговор: G = 'София' ;
         G = 'Берлин' ;
         G = 'Пловдив' ;
         G = 'Първомай' ;
         G = 'Силистра' ;
         G = 'Карнобат' ;
         no
```

11. Кои са градовете, чието население е по-малко от населението на някой друг град?

```
?- gr_nasel(G,N),gr_nasel(DrG,N1),N<N1.   или
?- gr_nasel(G,N),gr_nasel(DrG,N1),not N>=N1.
```

По-добре е с допълнителен предикат – `p(G):-gr_nasel(G,N),gr_nasel(_,N1),N<N1.`

Отговор: Всички градове без Абра Кадабра, който има най-голямо население. Има много повторения. Защо?

12. Като 11, но да се извеждат еднократно градовете.

```
Допълнителен предикат: p1(Gr):- gr_nasel(Gr,N),one (gr_nasel(_,N1),N<N1).
?- p1(M).
Отговор: M = 'София' ;
         M = 'Токио' ;
         M = 'Москва' ;
         M = 'Париж' ;
         M = 'Берлин' ;
         M = 'Пловдив' ;
         M = 'Първомай' ;
         M = 'Силистра' ;
         M = 'Карнобат' ;
         no
```

13. Кой град (градове) има най-малко население?

```
?- gr_nasel(Grad,N),\+ (gr_nasel(DrugGrad,N1),N>N1).
```

Не може да се използва not вместо \+, защото има свободни променливи в аргумента му.

```
Отговор: Grad = 'Първомай' ,
         N = 95000 ,
         DrugGrad = _ ,
         N1 = _ ;
         Grad = 'Силистра' ,
         N = 95000 ,
         DrugGrad = _ ,
         N1 = _ ;
         Grad = 'Карнобат' ,
         N = 95000 ,
```

```
              DrugGrad = _ ,
              N1 = _ ;
              no
    Или с допълнителен предикат: min(X):- gr_nasel(X,N),\+ (gr_nasel(_,N1),N>N1).
      ?- min(X).
      Отговор: N = 'Първомай' ;
               N = 'Силистра' ;
               N = 'Карнобат' ;
14. Кои са страните в Азия или Евроопа?
     ?-str_kont(X,'Азия');str_kont(X,'Европа').
     Отговор: X = 'Русия' ;
              X = Япония ;
              X = 'Русия' ;
              X = 'Франция' ;
              X = 'Магьосия'
*/
```

Зад. 2. В програма на Пролог са дефинирани предикатите person, male, female, likes, loves и has (виж по-долу в коментарите на програмата кой предикат какво отношение представя).

1. Дефинирайте предикати, описващи следните твърдения/отношения:

а) Щастлив е този, който има компютър и някой го обича.

б) Мария би харесала мъж, който я обича и има кола.

в) Предикат, който определя дали двама души харесват тенис.

г) Предикат, който определя, кой човек не харесва тенис.

д) Предикат, който определя кой човек не харесва нищо.

е) Предикат, който определя кога (дали) нещо се харесва поне от двама души.

ж) Мъж и жена могат да образуват двойка, ако се обичат и има поне едно нещо, което и двамата харесват.

з) Мъж и жена могат да образуват двойка, ако се обичат и харесват еднакви неща. (Всичко, което се харесва от единия, се харесва и от другия.)

2. Напишете вътрешни цели, които извеждат всички отговори на д) и ж).

```
/* програма people */
  % person(X) - X е човек
person(ivan).      person(petar).
person(maria).     person(anna).
person(elena).     person(dimitar).
person(tanya).     person(veselin).
  % male(X) - X е мъж
male(ivan).
male(petar).
male(dimitar).
  % female(X) - X е жена, ако е човек и не е мъж
  %(Всеки човек, който не е мъж, е жена.)
female(X):-person(X),not male(X).
  % likes(X,Y) - X харесва Y
likes(ivan,football).              ikes(ivan,computer_games).
likes(ivan,books).                 likes(maria,tennis).
likes(maria,flowers).              likes(anna,football).
likes(anna,books).                 likes(anna,computer_games).
likes(petar,tennis).               likes(tanya,tennis).
likes(maria,football).             likes(maria,books).
likes(dimitar,X):-likes(petar,X). % Димитър харесва всичко, което харесва Петър.
  % loves(X,Y) - X обича Y
loves(ivan,maria).  loves(ivan,anna).
loves(ivan,elena).  loves(petar,maria).
loves(elena,ivan).  loves(maria,petar).
loves(tanya,X):-male(X).  %Таня харесва всички мъже.
loves(dimitar,tanya).
loves(maria,ivan).
```

```
% has(X,Y) - X има Y
has(ivan,computer).   has(petar,car).
has(petar,computer).

% Щастлив е този, който има компютър и някой го обича.
happy(X):-has(X,computer),loves(_,X).

% Мария би харесала мъж, който я обича и има кола.
could_be_like(maria,X):-male(X),loves(X,maria),has(X,car).

% Предикат, който определя дали двама души харесват тенис.
like2tennis(X,Y):-likes(X,tennis),likes(Y,tennis),X\=Y.

% Предикат, който определя, кой човек не харесва тенис.
dont_like_tennis(Z):-person(Z),not like(Z,tennis).

% Предикат, който определя кой човек не харесва нищо.
likes_nothing(X):-person(X),\+ likes(X,_).

% Предикат, който определя кога нещо се харесва поне от двама души.
like2(X):-likes(A,X),likes(B,X),A\=B.
% Предикат, който определя кога нещо се харесва точно от двама души.
like2exactly(X):-likes(A,X),likes(B,X),A\=B,\+ (likes(Z,X),Z\=A,Z\=B).

% Мъж и жена могат да образуват двойка, ако се обичат и има поне едно нещо, което и
двамата харесват.
% Напишете предикат, който задава кои мъж и жена могат да образуват двойка.
pair(X,Y):-male(X),female(Y),loves(X,Y),loves(Y,X),one (likes(X,S),likes(Y,S)).

% Мъж и жена могат да образуват двойка, ако се обичат и харесват еднакви неща.
% (Всичко, което се харесва от единия, се харесва и от другия.)
% Напишете предикат, който задава кои мъж и жена могат да образуват двойка.
couple(X,Y):-male(X),female(Y),loves(X,Y),loves(Y,X),\+ (likes(X,S),not
likes(Y,S);likes(Y,S),not likes(X,S)).

%  Вътрешни цели
?-write('Кой не харесва нищо:'),nl,likes_nothing(X),write(X),nl,fail.
?-write('Възможни двойки:'),nl,pair(X,Y),write(X),write('--'),write(Y),nl,fail.
```

Вградени предикати:

- **! (отсичане)** – винаги успява. Използва се за предотвратяване на връщането назад за търсене на други решения (за преудовлетворяване на предишни подцели) или търсенето на решение по друга клауза.
- **not** p – успява, ако p не е истина. Аргументът не може да съдържа свободни променливи.
- **\+** p – успява, ако p не успее (p не е истина). Аргументът може да съдържа свободни променливи.

Внимание: Свободните променливи, участващи в аргумент на \+ не могат да получат стойност!

- **write(<аргумент>)** – отпечатва аргумента без да минава на нов ред.
- **nl –** преминаване на нов ред.
- **fail –** винаги пропада. Използва се когато искаме да предизвикаме връщане назад (за получаване на всички решения, цикъл, …). Понеже цел, в която участва fail винаги пропада, трябва ние сами да се погрижим да отпечатаме интересуващите ни стойности, получени като решение на подцелите преди fail. (Виж вътрешните цели в задачата.)

Когато има вътрешна цел, Пролог не отпечатва решения за променливите. За това трябва да се погрижи този, който пише програмата.

Упражнение 2 по Пролог (LPA Win-Prolog)

(8.11.2004 г.)

1. Съставни терми (структури) – представят съставни обекти (не функция, която връща резултат).
Записване – $f(a_1,a_2,..a_n)$; f – функтор, термите a_1, a_2, …, a_n – аргументи.
Терми са още променливите и константите.
Пример:

owns(tom,horse(blacky)).

Структурата е horse(blacky), представяща кон с неговото име. horse е функтора, а името blacky е аргумента.

owns(tom,car(mercedes,red).

2. Унификация на терми:
- константа (атомарен терм) се унифицира с равна на нея константа, свободна променлива или свързана променлива със стойност, равна на константата;
- променлива – ако е свободна се унифицира с всеки терм, ако е свързана – се унифицира по правилото за константи или съставни терми, в зависимост от стойността ѝ;
- съставен терм се унифицира само с такъв съставен терм, който има същия функтор и същия брой аргументи, и на който аргументите се унифицират със съответните аргументи на дадения терм.

3. Унификацията служи за:
- присвояване (свързване на променлива със стойност);
- предаване на параметри;
- проверка за равенство;
- достъп до структури от данни и техните елементи.

4. Примери за унификация на терми:
f(X,g(a,X),Y) и f(a,Y,g(Z,a)) се унифицират и в резултат на унификацията X=a, Y=g(a,a), Z=a.
g(s(X,Y),a,f(X)) и g(Z,Y,f(b)) се унифицират и Y=a, X=b, Z=s(a,b)
f(a,g(X),s(X)) и f(Y,g(c),s(b)) не се унифицират
f(a,g(X),s(b)) и f(Y,g(c),s(b)) се унифицират и X=c, Y=a

Задача 1. Дефинирайте предикат, който задава кой човек какво притежава. Нещата, които могат да се притежават са: банкова сметка с определена сума; апартамент или къща, които се характеризират с местоположение, площ и цена; кола от определена марка и с даден цвят; яхта.
Напишете:

1. Цел, извеждаща какво има Иван.
2. Цел, извеждаща кой има кола.
3. Цел, извеждаща кой каква марка кола има.
4. Цел, установяваща има ли двама с червени коли.
5. Цел, установяваща има ли двама с еднакъв цвят коли.
6. Цел, извеждаща кой има в банковата си сметка толкова пари, колкото струва домът му.
7. Предикат, определящ кой какъв дом има (домът е цяла структура) и цел, извеждаща кой какъв дом има.
8. Предикат, определящ големите домове - с площ повече от 200 кв. м., и техните собственици и цел, извеждаща кой има голям дом.
9. Предикат, задаващ кой е богат - ако има къща или апартамент на цена повече от 30000, яхта, кола и банкова сметка над 200000 и цел, извеждаща кой е богат?
10. Предикат, определящ колко е най-голямата банкова сметка и цел, която я извежда.
11. Предикат, определящ жилището с най-голяма площ и цел, която го извежда. (Да се използва допълнителен предикат, определящ кое е жилище.)

```
/* програма owner */

/* База факти с данни за това кой какво има */
has(ivan,yacht).                  has(ivan,car(lada,red)).
has(tanya,bank_account(200000)). has(peter,house(plovdiv,space(160),price(48000))).
has(maria,flat(sofia,space(121),price(32000))).
has(kamen,car(renault,blue)).     has(kamen,bank_account(1005000)).
has(ivan,flat(varna,space(110),price(60000))).
has(ivan,bank_account(2000000)). has(maria,bank_account(600000)).
has(maria,yacht).                 has(elena,car(ford,red)).
has(maria,house(tzarevo,space(300),price(600000))).
has(maria,car(volvo,black)).      has(peter,bank_account(2000000)).
/*-----------------------------------------------------------------*/
rich(X):- has(X,yacht), has(X,car(_,_)), has(X,bank_account(Z)), Z>200000,
          one ((has(X,house(_,_,price(Y))); has(X,flat(_,_,price(Y)))), Y>30000).
biggest_account(Z):- has(_,bank_account(Z)), \+ (has(_,bank_account(C)), C>Z),!.
home(X,Y):- has(X,Y),(Y=flat(_,_,_);Y=house(_,_,_)).
big_home(X,Y):- has(X,Y),(Y=flat(_,space(Z),_);Y=house(_,space(Z),_)),Z>200.
ishome(X):- has(_,X),(X=flat(_,_,_);X=house(_,_,_)).
biggest_home(X):-ishome(X),X=_(_,space(Z),_), \+ (ishome(_(_,space(Y),_)),Y>Z).
/*
1. Какво има Иван?
       ?- has(ivan,X).
2. Кой има кола?
       ?- has(X, car(_,_)). или ?- has(X,Y),functor(Y,car,_).
3. Кой каква марка кола има?
       ?- has(X,car(Y,_)).
4. Има ли двама с червени коли?
       ?-has(X,car(_,red)),has(Y,car(_,red)),X\=Y,!.
5. Има ли двама с еднакъв цвят коли?
       ?- has(X,car(_,Z)),has(Y,car(_,Z)),X\=Y,!.
6. Кой има в банковата си сметка толкова пари, колкото струва домът му?
       ?- has(X,bank_account(Y)),
          (has(X, flat(_,_,price(Y))); has(X, house(_,_,price(Y)))).
7. а) Предикат, определящ кой какъв дом има (домът е цяла структура).
       home(X,Y):- has(X,Y),(Y=flat(_,_,_);Y=house(_,_,_)).
   б) Кой какъв дом има?
       ?- home(X,U).
8. а) Предикат определящ големите домове - с площ повече от 200 кв. м.,
       и техните собственици.
       big_home(X,Y):- has(X,Y),(Y=flat(_,space(Z),_);Y=house(_,space(Z),_)),Z>200.
       или big_home(X,Y):- has(X,Y),Y=_(_,space(Z),_),Z>200.
   б) Кой има голям дом?
       ?- big_home(X,_).
9. а) Предикат, задаващ кой е богат - ако има къща
       или апартамент на цена повече от 30000, яхта,
       кола и банкова сметка над 200000.
       rich(X):- has(X,yacht), has(X,car(_,_)), has(X,bank_account(Z)), Z>200000,
              one ((has(X,house(_,_,price(Y))); has(X,flat(_,_,price(Y)))), Y>30000).
      (извежда по веднъж всеки богат - one)
   б) Кой е богат?
      ?- rich(X).
10. а) Предикат, определящ колко е най-голямата банкова сметка.
     biggest_account(Z):- has(_,bank_account(Z)), \+ (has(_,bank_account(C)), C>Z),!.
    б) Колко е най-голямата сметка?
      ?- biggest_account(X).
11. Да се намери жилището с най-голяма площ. Да се използва допълнителен
   предикат, определящ кое е жилище.
   ishome(X):- has(_,X),(X=flat(_,_,_);X=house(_,_,_)).
   biggest_home(X):-ishome(X),X=_(_,space(Z),_), \+ (ishome(_(_,space(Y),_)),Y>Z).
   ?- biggest_home(X).
*/
```

Задача 2. Дадени са факти на Пролог, описващи кой е автора на дадена книга и каква е цената й . Книгата е структура от вида book(Заглавие,Автор,Брой_страници), а авторът – структура author(Собствено_име, Фамилия). Дефиниран е и предикат, задаващ кой автор от каква националност е.

1. Да се изведат всички книги с цена над 3.50.
2. Да се изведат всички книги с повече от 180 страници:
   а) книгите да се изведат с отделни променливи за име, автор и страници;
   б) книгите да се изведат като цели структури.
3. Да се намери автор с две различни книги.
4. Да се намери автор с точно една книга.
5. Да се намерят всички двойки различни автори с едно и също първо име.
6. Да се изведат заглавията на книгите на американските автори.
7. Да се намери най-евтината книга.

```
/* програма publications */

/* Автор – структура author(Собствено име, Фамилия)
   Книга – структура book(Заглавие,Автор,Брой страници)
   Публикация – предикат publication(Книга,Цена)
   Националност – предикат nationality(Автор,Националност на автора)
  */
publication(book('Програмиране на Пролог',author('Данаил','Дочев'),247),2.89).
publication(book('Прослава в смъртта',author('Нора', 'Робъртс'),351),3.50).
publication(book('Безсмъртие в смъртта',author('Нора', 'Робъртс'),367),4.20).
publication(book('Полетът на интрудър',author('Стивън','Кунц'),320),3.40).
publication(book('След раздялата',author('Ги','Шантльор'),152),2.10).
publication(book('Шепот в здрача',author('Барбара','Делински'),391),4.80).
publication(book('Неандерталец',author('Джон','Дарнтън'),350),3.90).
publication(book('Скватерите',author('Майн','Рид'),400),4.00).
nationality(author('Данаил','Дочев'),'българин').
nationality(author('Нора','Робъртс'),'американец').
nationality(author('Барбара','Делински'),'американец').
nationality(author('Ги','Шантльор'),'французин').
nationality(author('Джон','Дарнтън'),'американец').

two_books(X):-publication(book(Y,X,_),_),publication(book(Z,X,_),_),Z\=Y.
one_book(X):- publication(book(Y,X,_),_), \+ (publication(book(Z,X,_),_),Z\=Y).
same_fname(X,Y):-publication(book(_,X,_),_),publication(book(_,Y,_),_),
                 X\=Y,X=author(Z,_),Y=author(Z,_).
/*
1. Да се изведат всички книги с цена над 3.50.
?- publication(X,Y),Y>3.50.
2. Да се изведат всички книги с повече от 180 страници:
   а) книгите да се изведат с отделни променливи за име, автор и страници;
?- publication(book(X,Y,Z),_),Z>180.
   б) книгите да се изведат като цели структури.
?- publication(X,_),X=book(_,_,Z),Z>180,write(X),nl,fail.
3. Да се намери автор с две различни книги.
two_books(X):-publication(book(Y,X,_),_),publication(book(Z,X,_),_),Z\=Y.
?- two_books(X),!.
4. Да се намери автор с точно една книга.
one_book(X):- publication(book(Y,X,_),_), \+ (publication(book(Z,X,_),_),Z\=Y).
?-one_book(X),!.
5. Всички двойки различни автори с едно и също първо име.
same_fname(X,Y):-publication(book(_,author(Z,X),_),_),
                 publlication(book(_,author(Z,Y),_),_),X\=Y.
   или
same_fname(X,Y):-publication(book(_,X,_),_),publication(book(_,Y,_),_),
                 X\=Y,X=author(Z,_),Y=author(Z,_).
?- same_fname(X,Y).
```

```
6. Заглавията на книги на американски автори.
?- publication(book(X,Y,_),_),nationality(Y,'американец'),write(X),nl,fail.
7. Да се намери най-евтината книга.
?-publication(X,Y),\+ (publication(_,Z),Z<Y).
*/
```

Задачи за упражнение:

Задача 1. Напишете факти на Пролог, задаващи кой ВУЗ в кой град се намира и факти, задаващи кой студент в кой ВУЗ учи и на каква възраст е. Напишете цели, получаващи отговор на въпросите:

а) Кои са ВУЗ-овете в София?

б) Кои са студентите, учещи в Пловдив?

в) В кои градове има студенти по-възрастни от 30 г.? Да се извеждат името на студента и името на града.

Задача 2. Дадена е база факти на Пролог от следния вид:

```
% work_at(лекар,лечебно_заведение,трудов_стаж)
work_at('Иван Петров','Трета поликлиника',23).
work_at('Михаела Куин','Колорадо Спрингс',8).
work_at('Стоян Драганов','Първа поликлиника',14).
…
% heals(лекар,пациент) – описващ отношението лекуващ лекар – пациент;
% пациент е структура patient(име,възраст)
heals('Михаела Куин', patient('Съли',34)).
heals('Стоян Драганов', patient('Нели Томова',53)).
heals('Иван Петров', patient('Камен Димов',23)).
heals('Стоян Драганов', patient('Христо Ангелов',76)).
heals('Стоян Драганов', patient('Камен Димов',23)).
…
```

а) Дефинирайте предикат, който определя лекарите с трудов стаж 10 години.

б) Напишете цел, която извежда като цели структури пациентите на възраст 34 г., лекувани от д-р Михаела Куин.

в) Дефинирайте предикат, определящ имената на пациентите, които се лекуват при лекар, работещ в Трета поликлиника и имащ под 15 години трудов стаж.

г) Напишете цел, която извежда двама различни пациенти (като структури), които се лекуват при един и същи лекар.

д) Дефинирайте предикат, който определя пациентите (като цели структури), които не се лекуват в Първа поликлиника.

е) Напишете цел, която извежда пациентите, които се лекуват само при един лекар.

Задача 3.

а) Напишете факти, описващи кое момиче от кой град е и на каква възраст е;

б) Напишете факти, описващи кое момиче колко е високо и какви са мерките му. Мерките се представят чрез структура с 3 аргумента – гръдна обиколка, талия, ханш;

в) Напишете правило, определящо кое момиче става за манекен – това, което има ръст над 172 см и е под 25 години.

г) Задайте цел, която извежда момичетата, които са на възраст под 17 г.;

д) Задайте цел, която извежда мерките на момичетата от Пловдив с ръст под 170 см.;

е) Задайте цел, която извежда град, от който няма нито едно момиче, на възраст 20 г.;

ж) Задайте цел, която проверява дали се срещат две различни момичета с еднаква гръдна обиколка.

Упражнение 3 по Пролог (LPA Win-Prolog)

(15.11.2004 г.)

I. Аритметични терми.

А/ Аритм. изрази - специален вид терми, които могат да се изчисляват.

-A*B+2*C = '+'('-'('*'(A,B)),'*'(2,C)).

(знакът за операция е функторът на терма)

Операции: +, -, *, /, // (целочислено деление), mod

Стандартни функции: abs, sign, int, sqrt, ln, log, sin, cos, tan, …

Сравняване на аритм. изрази: <, =<, >, >=, =:= (срвн. за равенство), =\= (за неравенство)

Б/ Разлика между =, \=, is, =:=, =\=, ==, \==.

терм1 **=** терм2 – (проверка за) унификация на терм1 и терм2;

терм1 **\=** терм2 – проверка за неунификация (истина е, ако не се унифицират двата терма);

ар.изр.1 **=:=** ар.изр.2 – сравняване на аритм. изрази за равенство, като аритм. изрази от двете страни се изчисляват;

ар.изр.1 **=\=** ар.изр.2 – сравняване на аритм. изрази за неравенство;

пром. **is** ар.изр. – свързване на свободна променлива със стойността (изчислява се стойността на израза); променливата може да е и свободна – тогава се сравняват стойностите от двете страни;

терм1 **==** терм2 – проверка за идентичност (съвпадение);

терм1 **\==** терм2 – проверка за неидентичност.

Примери:

```
?- a(1,X) = a(1,2).
X = 2
?- a(1,X) == a(1,2).
no
?- X = 2, a(1,X) == a(1,2).
X = 2
```

```
?- 2+3 = 1+4.
no
?- 2+3 =:= 1+4.
yes
?- X = 1+4.
X = 1 + 4
```

```
?- X is 1+4.
X = 5
?- X = 5, X is 1+4.
X = 5
```

Задача: Дефинирайте предикат за пресмятане по зададено x на стойността на функциите:

$$g(x)=\begin{cases}\ln(x+3), & 0{,}5 < x \le 5\\ -3, & -0{,}5 \le x \le 0{,}5\\ \sqrt{|12-x|}+x^2, & x < -0{,}5 \text{ или } x > 5\end{cases} \quad \text{и} \quad f(x)=\begin{cases}\ln|x+1|, & x \le 1 \text{ и } x \ne -1\\ x^3+\dfrac{3}{x}, & x > 1 \text{ и } x \ne 5\\ x, & x = -1 \text{ или } x = 5\end{cases}.$$

```
g(X,Y):- X>0.5,X=<5,!, Y is ln(X+3).
g(X, -3):- X>= -0.5, X=<0.5,!.
g(X,Y):- Y is sqrt(abs(12-X))+X*X.
```

```
f(-1,-1):- !.
f(5,5):- !.
f(X,Y):- X=<1, !, Y is ln(abs(X+1)).
f(X,Y):- Y is X*X*X+3/X.
```

II. Рекурсия.

Рекурсивен предикат – в дясната част на някое негово правило се среща същият предикат.

p(X):-a(X,Y),p(Y).

Използване:

- задачата се формулира рекурсивно (отношението се описва чрез себе си);
- обработва се рекурсивна структура от данни.

Рекурсията – единствен начин за реализиране на итерация (цикли).

Рекурсивната дефиниция включва:

- стоп-клауза (една или повече) за завършване на рекурсията
- рекурсивна клауза. (една или повече)

Задачи:

1. Дефиниране на потомък (наследник) чрез предикатите син и дъщеря.

```
descendent(X,ivan):-son(X,ivan).                          % стоп-клауза
descendent(X,ivan):-daughter(X,ivan).                     % стоп-клауза
descendent(X,ivan):-son(Y,ivan), descendent(X,Y).         % рекурсивна клауза
descendent(X,ivan):-daughter(Y,ivan), descendent(X,Y).    % рекурсивна клауза
```

2. Пресмятане на факториел.

а) Рекурсивен вариант – съответства на написване на рекурсивна подпрограма на процедурен език. Това е естественият начин на дефиниране на отношения – виж горе предиката descendent:

```
fact(0,1).                                  % 0!=1          (стоп-клауза)
fact(N,F):- N>0, N1 is N-1, fact(N1,F1), F is F1*N.   % n!=n*(n-1)!   (рекурсивна клауза)
```

(Добавяне на ! в първата клауза (fact(0,1):-!.); махане на сравнението N>0 във втората?)

б) Итеративен вариант – съответства на написване на цикъл на процедурен език. Предикатът е пак рекурсивен, но рекурсивното извикване е последно в последната клауза (дясна рекурсия), което позволява по-голяма ефективност:

```
fact(N,F):-f(0,N,1,F).                          |  x=0; f=1;
f(N,N,F,F).                                     |  while (x < n)
f(X,N,Y,F):- X<N, X1 is X+1, Y1 is Y*X1, f(X1,N,Y1,F).  |  { x++; f=f*x; }
```

Внимание! Ако един предикат се извика с аргумент свързана променлива, не може тази променлива да приеме друга стойност в процеса на търсене на решение.
Не може да присвоявате стойност така: X is X+1, защото от двете страни на is трябва да има еднакви стойности, а никое число не е равно на самото то плюс 1. Т. е. щом променлива е свързана с една стойност, тя не може да се свърже чрез присвояване (is) с друга стойност.

3. Пресмятане на степен: x^n, n>0.

I начин

```
stepen(X,1,X):-!.                                  %  x^1=x
stepen(X,N,Y):- N>1, N1 is N-1, stepen(X,N1,Y1), Y is Y1*X.   %  x^n=x*x^(n-1)
```

II начин

```
step(X,N,Y):- st(X,N,1,X,Y).
st(_,N,N,Y,Y):-!.
st(X,N,I,P,Y):- I<N, I1 is I+1, P1 is P*X, st(X,N,I1,P1,Y).
```

4. Отпечатване на екрана на числата от 1 до зададено число.

I начин

```
w(0).
w(X):-X>0, Y is X-1, w(Y),write(X),nl.
```

II начин

```
wrt(N):-w1(N,1).
w1(N,I):-I=<N,!,write(I),nl,I1 is I+1,w1(N,I1).
w1(_,_).
```

5. Пресмятане по дадени *x* и *n* на сумата $(x+2)^2+(x+4)^2+(x+6)^2+\ldots+(x+a)^2$, където *a* е най-голямото четно число, ненадхвърлящо *n*.

I начин

```
pred(X,N,S):- N>0, (N mod 2 =:=1, !, N1 is N-1, pr(X,N1,S) ; pr(X,N,S)).
pr(_,0,0):- !.
pr(X,N,S):- N1 is N-2, pr(X,N1,S1), S is S1+(X+N)*(X+N).
```

II начин

```
predic(X,N,S):- cycle(X,N,S,2,0).
cycle(_,N,S,I,S):- I>N,!.
cycle(X,N,S,I,Y):- Y1 is Y+(X+I)*(X+I), I1 is I+2, cycle(X,N,S,I1,Y1).
```

6. Пресмятане на НОД на две числа..

```
nod(X,0,X):-!.                          % или nod(X,Y,Y):- X mod Y =:= 0, !.
nod(X,Y,D):-R is X mod Y, nod(Y,R,D).
```

7. Пресмятане на *n*-ти елемент на редицата на Фибоначи: $f_0=1;\ f_1=1;\ f_n=f_{n-1}+f_{n-2},\ n>1$.

I начин

```
fib(0,1).
fib(1,1).
fib(N,X):- N>1, Y is N-1, Z is N-2, fib(Y,X1),fib(Z,X2), X is X1+X2.
```

II начин

```
fib1(N,X):-fi(0,0,1,N,X).
fi(N,_,X,N,X):-!.
fi(I,S1,S2,N,X):- I<N, I1 is I+1, S is S1+S2, fi(I1,S2,S,N,X).
```

8. Пресмятане на сумата на безкраен ред със зададена точност: $x-x^3/3!+x^5/5!-x^7/7! \ldots = \sin(x)$

```
suma(X,S,Eps):-s(X,X,1,X,S,Eps).
s(X,X1,I,S1,S,Eps):- abs(X1)>=Eps,!,J is I+2, X2 is -X*X*X1/(J*(I+1)), S2 is S1+X2, s(X,X2,J,S2,S,Eps).
s(_,_,_,S,S,_).
```

9. Имаме факти от вида: age(sara,11). age(liza,11). age(petar,9). age(uri,49). age(dimo,52).
Да се напише предикат за отпечатване на всички хора, които са на възраст от X до Y години в ред на нарастване на възрастта.

I начин

```
w(X,Y):-X=<Y,age(I,X),write(I),nl,fail.
w(X,Y):-X=<Y,!,X1 is X+1,w(X1,Y).
w(_,_).
```

Каква е ролята на последната клауза на w?

Друг начин е с помощен предикат, осъществяващ получаването на стойностите от X до Y (свързването на променлива с тези стойности).

II начин

```
ww(I,I,_).
ww(I,X,N):- X<N, X1 is X+1, ww(I,X1,N).
p(X,Y):- ww(I,X,Y), age(N,I), write(N), write(' - '), nl, fail.
```

10. Дефинирайте предикат за въвеждане в цикъл на аритметични изрази (до въвеждане на quit) и отпечатване на тяхната стойност.

```
eval:- repeat, write('Input expression: '), nl, read(X), (X=quit, ! ; Y is X, write(Y), nl, fail).
```

<u>Задачи за упражнение:</u>

1. Дефинирайте предикати за намиране на стойността на функциите:

$$f(x,y)=\begin{cases}\sqrt{\left|y^2-3.x^2\right|}, & y>x\\ x.\sin x, & x=y\\ e^x+\ln(x-y), & x>y\end{cases}, \qquad h(y)=\begin{cases}\dfrac{2.y^2}{y+4}, & y<-4\\ 0, & y\in[-4,4]\\ \dfrac{\ln(y-4).e^y}{y^3+\sqrt{y}}, & y>4\end{cases} \quad \text{и} \quad k(n)=\frac{n!}{(n+2)^2}.$$

2. Дефинирайте предикат за пресмятане на сумата: $1+\frac{1}{2^2}+\frac{1}{3^2}+\ldots+\frac{1}{n^2}$ по дадено n.

3. Дефинирайте предикат за пресмятане на сумата: $1+\frac{1}{2}.x^2+\frac{1.3}{2.4}.x^4+\frac{1.3.5}{2.4.6}.x^6+\ldots$ за $|x|<1$ със зададена точност – много малко ε (абсолютна грешка) .

4. Дефинирайте предикат за пресмятане на произведението $n.(n-k).(n-2k)\ldots(n-k^2)$ при дадени n и k.

5. Дефинирайте предикат за пресмятане на произведението $\frac{1}{2^2-1}.\frac{1}{3^2-2}\ldots\frac{1}{n^2-n+1}$ по дадено n.

6. Дефинирайте предикат, който намира n-тия член на рекурентната редица $T_0(x)=1$, $T_1(x)=x$, $T_k(x)=2.x.T_{k-1}(x)-T_{k-2}(x)$, $k = 2, 3, \ldots$ при зададени x и n.

7. Дадена е редицата $a_n=\frac{n!}{5^n}, n=1,2,\ldots$ Дефинирайте предикат, който намира първото a_n, което е по-голямо от дадено x (т.е. $a_n > x$ и n - минимално).

Упражнение 4 по Пролог (LPA Win-Prolog)

(22.11.2004 г.)

1. Списък – наредена последователност от 0, 1 или повече елементи. Елементите могат да бъдат произволни терми, включително други списъци. (Списъците също са терми – специален вид структури.)

Примери:

[] – празен списък
[1] – списък с един елемент
[a,b] – списък с два елемента
[p(maria,ivan),3] – списък с два елемента
[a,b,f(N,k),1,[a,1]] – списък с 5 елемента

2. Обработка на списъци – чрез отделяне на глава и опашка.

Глава - първият елемент на списъка

Опашка - списък от елементите без първия

Списък	Глава	Опашка
[a,b]	a	[b]
[1]	1	[]
[]	няма	няма
[X,f(Y),1]	X	[f(Y),1]
[[1,2],[3,4]]	[1,2]	[[3,4]]
[X+Y,p(sofia)]	X+Y	[p(sofia)]

Записът [X|Y] означава списък с глава X и опашка Y. При унификация на даден списък с [X|Y], променливата X се свързва с главата на дадения списък, а променливата Y – с опашката му. Така е възможен достъп до елементите и обработка на списък.

3. Унификация на списъци – примери.

```
?- [X|Y]=[a,b,c].
X = a , Y = [b,c]
?- [X,Y|Z]=[a,b].
X = a ,Y = b , Z = []
?-  [X,Y|Z]=[a].
no
?-  [X,_,Y|Z]=[a,b,c,d,e].
X = a ,Y = c , Z = [d,e]
?- [X,Y]=[a,b,c,d].
no
?-  [X,Y]=[a,b].
X = a , Y = b
?- [X,a,X,f(X,a)|Y]=[Z,Z|L].
X = Z = a , Y = _ , L = [a,f(a,a)|Y]
?- [_,X,7,Y,_,f(1,abc)|Z]=[99,abc,ABC,X,f(s,s,s),C,3,abc|_].
X = Y = abc , Z = [3,abc|_] , ABC = 7 , C = f(1,abc)
?- [X,7,[Y|L],34|Z]=[[a,b,c],ABC,X,34,a(1,2)|[x,y,z]].
X = [a,b,c] , Y = a , L = [b,c] , Z = [a(1,2),x,y,z] , ABC = 7
```

4. Дефинициите на предикати, обработващи списъци са обикновено *рекурсивни*. Такава дефиниция описва непосредствено характеристиките (обработката) на главата на списъка или на първите няколко елемента, а характеристиките (обработката) на опашката се описват чрез рекурсивно използване на същия предикат.

5. Отношение между терм и списък.

<терм> =.. <списък> – списъкът е с глава функтора на терма и опашка списъка от аргументи на терма.

Примери:

```
?- g(tom,cat,5) =.. [X|Y].
X = g , Y = [tom,cat,5]
```

```
?- Y=..[a,1,2,v,e].
Y = a(1,2,v,e)
```

Задачи:

1. Дефинирайте предикат, който задава съответствието между списък и неговата дължина (броя на елементите му). Еквивалентен е на вградения предикат **length(<списък>,<дължина>)**.

```
length1([],0).
length1([_|T],X):-length1(T,Y), X is Y+1.
```

2. Дефинирайте предикат, който описва принадлежността на елемент към списък. Еквивалентен е на вградения предикат **member(<елемент>,<списък>)**.

```
memb(X,[X|_]).
memb(X,[_|T]):-memb(X,T).
```

Каква е разликата между memb и предиката memb1, дефиниран по-долу?

```
memb1(X,[X|_]):-!.
memb1(X,[_|T]):-memb1(X,T).
```

Задайте целите: ?- memb(X,[1,2,3,1,4,1]).
?- memb1(X,[1,2,3,1,4,1]).
?- X=1,memb(X,[1,2,3,1,4,1]).
?- X=1,memb1(X,[1,2,3,1,4,1]).

3. Дефинирайте предикат, който преброява колко пъти даден терм се среща в даден списък.

```
brX(_,[],0).
brX(X,[X|T],B):-!,brX(X,T,B1), B is B1+1.
brX(X,[_|T],B):-brX(X,T,B).
```

4. Дефинирайте предикат, с който може да се преброява колко пъти в даден списък се среща структура st(X,Y,Z) с втори аргумент (т.е. Y) число.

```
br([],0).
br([st(_,Y,_)|T],B):-number(Y),!,br(T,C),B is C+1.
br([_|T],B):-br(T,B).
```

5. Дефинирайте предикат, който определя сумата на елементите на числов списък.

```
suma([],0).
suma([H|T],S):-suma(T,S1), S is S1+H.
II н.
sum(L,S):-su(L,0,S).
su([],S,S).
su([H|T],S1,S):-S2 is S1+H,su(T,S2,S).
```

6. Дефинирайте предикат, за пресмятане на сумата от произведенията на съответните елементи на два списъка.

```
su([H1|T1],[H2|T2],S):- !, su(T1,T2,S1), S is S1 +H1*H2.
su(_,_,0).
```

7. Дефинирайте предикат, за пресмятане на сумата от елементите на четно място в списък.

```
ssuma([],0):- !.
ssuma([_],0):-!.
ssuma([_,X|Y],S):- ssuma(Y,S1), S is S1+X.
```

8. Дефинирайте предикат, за пресмятане на стойността на полинома $a_n.x^n+a_{n-1}.x^{n-1}+\ldots+a_1.x+a_0$ по схемата на Хорнер. (Коефициентите пред неизвестното се подават чрез списък.)

```
poli(X,[H|T],P):- p(X,T,P,H).
p(_,[],P,P):-!.
p(X,[H|T],P,Y):- Y1 is Y*X+H,p(X,T,P,Y1).
```

9. Дефинирайте предикат, който проверява дали термите X и Y са съседни елементи в списък.

```
s(X,Y,[X,Y|_]):-!.
s(X,Y,[Y,X|_]):-!.
s(X,Y,[_|L]):-s(X,Y,L).
```

10. Дефинирайте предикат, с който може да се намира последният елемент на списък.

```
last(X,[X]):-!.
last(X,[_|T]):-last(X,T).
```

11. Дефинирайте предикат, определящ N-ия елемент в даден списък при известно N.

```
nmem(1,[X|_],X):-!.
nmem(N,[_|T],X):-N>1,N1 is N-1,nmem(N1,T,X).
```

12. Дефинирайте предикат, определящ позицията на първото срещане на даден терм в даден списък.

```
pos(1,[X|_],X):-!.
pos(N,[_|T],X):-pos(N1,T,X),N is N1+1.
```

13. Дефинирайте предикат, определящ позициите (всички) в даден списък, в които се среща даден терм.

```
posmem(X,[X|_],1).
posmem(X,[_|T],N):-posmem(X,T,N1),N is N1+1.
```

Този предикат работи и в двете посоки – и позицията, и елемента могат да бъдат неизвестни.

Еквивалентен е на вградения предикат **member(<елемент>, <списък>, <позиция>)**, определящ позициите в списък, в които се среща даден терм. Също може да определи елемента, който се намира на дадена позиция.

14. Дефинирайте предикат, който определя дали един списък е получен чрез конкатенацията на други два списъка. Аналог е на вградения **append(<списък1>,<списък2>,<конкатенация на списък1 и списък2>)**.

```
concat([],S,S).
concat([X|L],S,[X|T]):-concat(L,S,T).
```

15. Дефинирайте предикат, който намира сумата на два N-мерни вектора, като всички вектори се представят чрез списъци от числа.

```
add([H|T],[K|L],[M|R]):-M is H+K,add(T,L,R).
add([],[],[]).
```

16. Дефинирайте предикат, чрез който може да се вмъкне даден терм, на N-то място в даден списък.

```
ins(1,X,T,[X|T]).
ins(N,X,[H|T],[H|L]):-N>1,N1 is N-1,ins(N1,X,T,L).
```

17. Дефинирайте предикат, чрез който може да се премахне първото срещане на даден елемент в списък.

```
del(_,[],[]):-!.
del(X,[X|T],T):-!.
del(X,[Y|T],[Y|L]):-del(X,T,L).
```

Има вграден предикат **remove(<елемент>, <списък>, <нов списък>)**, който успява толкова пъти, колкото пъти се среща премахваният елемент в началния списък, и при всеки успех премахва само по едно участие. Например remove(2,[1,2,3,2,4],X). успява два пъти съответно за X=[1,3,2,4] и за X=[1,2,3,4].

18. Дефинирайте предикат, чрез който може да се премахнат едновременно всички участия на даден елемент в даден списък. Аналог е на вградения предикат **removeall(<елемент>, <списък>, <нов списък>)**.

```
delall(_,[],[]):-!.
delall(X,[X|T],T1):-!,delall(X,T,T1).
delall(X,[Y|T],[Y|L]):-delall(X,T,L).
```

19. Дефинирайте предикат, който проверява дали един списък се получава от друг чрез замяна на всеки елемент на първият списък, равен на даден терм A, с терма B.

```
subst(_,_,[],[]).
subst(A,B,[A|T],[B|L]):-!,subst(A,B,T,L).
subst(A,B,[X|T],[X|L]):-subst(A,B,T,L).
```

20. Дефинирайте предикат, който от даден числов списък генерира два списъка: единият, съдържащ елементите на дадения списък, които са по-малки от дадено число X, а другият, съдържащ всички останали елементи на първия списък.

```
split(_,[],[],[]).
split(X,[H|T],[H|T1],T2):-H<X,!,split(X,T,T1,T2).
split(X,[H|T],T1,[H|T2]):-split(X,T,T1,T2).
```

21. Дефинирайте предикат, който инвертира списък.

Аналог е на вградения **reverse(<списък>, <инвертиран списък>)**.

```
invert([],[]).
invert([H|T],L):-invert(T,T1),append(T1,[H],L).
II н.
rev(L,L1):-r(L,[],L1).
r([],L,L).
r([H|T],L,R):-r(T,[H|L],R).
```

22. Дефинирайте предикат, който описва кой списък е симетричен, т.е. първият му елемент е равен на последния, вторият - на предпоследния и т.н.

```
simetr(L):-reverse(L,L).
II н.
palindrom(L):-pal(L,[]).
pal(L,L):-!.
pal([_|L],L):-!.
pal([H|T],L):-pal(T,[H|L]).
```

23. Дефинирайте предикат, който описва кога един списък е подсписък на друг съответно:

а) подсписък от последователни елементи;

```
podsp(X,L):- append(Y,_,L),append(_,X,Y),!.
```

II н.

```
podspis([H|T],[H|L]):-append(T,_,L).
podspis(X,[_|L]):-podspis(X,L).
podspis([],_).
```

б) "подсписък" (подредица) от непоследователни елементи.

```
subser([H|T],[H|L]):- subser(T,L).
subser(X,[_|L]):- subser(X,L).
subser([],_).
```

24. Дефинирайте предикат, описващ кой елемент на даден числов списък е минимален (максимален).

```
min([M],M):-!.
min([H|T],M):-min(T,M),M<H,!.
min([H|_],H).
```

25. Дефинирайте предикат, който на даден числов списък съпоставя получения от него сортиран списък. Има вграден предикат **sort(<списък>,<сортиран списък без повторения>)**.

Сортировка - мехурче

```
bubble(L,R):-change(L,L1),!,bubble(L1,R).
bubble(L,L).
change([X,Y|T],[Y,X|T]):-X>Y.
change([X|T],[X|L]):-change(T,L).
```

Сортировка - чрез избор на минимален елемент

```
sortsel([],[]).
sortsel([H|T],[M|L]):-minel(H,M,T,R),!,sortsel(R,L).
minel(M,M,[],[]).
minel(X,M,[H|T],[X|L]):-H<X,!,minel(H,M,T,L).
minel(X,M,[H|T],[H|L]):-minel(X,M,T,L).
```

Сортировка - вмъкване

```
sortins([],[]).
sortins([H|T],L):-sortins(T,R),insert(H,R,L).
insert(X,[H|T],[H|L]):-H<X,!,insert(X,T,L).
insert(X,L,[X|L]).
```

II н.

```
sortins1(L,R):-sort2(L,[],R).
sort2([H|T],L,R):-insert(H,L,L1),sort2(T,L1,R).
sort2([],R,R).
```

Бърза сортировка

```
sortq([],[]).
sortq([H|T],L):-split(H,T,T1,T2),sortq(T1,L1),sortq(T2,L2),append(L1,[H|L2],L).
```

26. Дефинирайте предикат, който успява, ако даден списък представя множество, т.е. няма повтарящи се елементи.

```
set([]).
set([H|T]):-not member(H,T),set(T).
```

27. Дефинирайте предикат, който на даден списък съпоставя списъка, получен от дадения чрез премахване на повторенията на елементи.

```
rem_dupl([],[]).
rem_dupl([H|T],[H|L]):-removeall(H,T,T1),rem_dupl(T1,L).
```

II н.
```
del_dupl([],[]).
del_dupl([H|T],[H|L]):-not member(H,T),!,del_dupl(T,L).
del_dupl([_|T],L):-del_dupl(T,L).
```

28. Дефинирайте предикат, който описва кога един списък е подмножество на друг.
```
subset([],_).
subset([H|T],L):-member(H,L),subset(T,L).
```

29. Дефинирайте предикат за намиране на сечението (обединението/ разликата) на две множества, представени чрез списъци.

Сечение
```
intersect([],_,[]).
intersect([H|T],L,[H|R]):-member(H,L),!,intersect(T,L,R).
intersect([_|T],L,R):-intersect(T,L,R).
```
Обединение
```
union([],L,L).
union([H|T],L,[H|R]):- not member(H,L),!,union(T,L,R).
union([_|T],L,R):-union(T,L,R).
```
Разлика
```
difference([],_,[]).
difference([H|T],L,[H|R]):- not member(H,L),!,difference(T,L,R).
difference([_|T],L,R):-difference(T,L,R).
```

30. Дефинирайте предикат, който намира броя на елементите на числов списък, които са равни на позицията, в която се намират.
```
elpos(L,B):-brpos(L,L,B).
brpos([X|T],L,B):- not member(X,T),member(X,L,X),!,brpos(T,L,B1), B is B1+1.
brpos([_|T],L,B):-brpos(T,L,B).
brpos([],_,0).
II н.
c(L,B):-c1(L,1,B).
c1([X|T],X,B):-!,N is X+1,c1(T,N,B1), B is B1+1.
c1([_|T],N,B):-M is N+1, c1(T,M,B).
c1([],_,0).
```

31. Дефинирайте предикат, който заменя с f(X) всички елементи X на списъка L, които не са елементи на списъка L1.
```
zam([],_,[]).
zam([X|Y],L,[f(X)|Y1]):- not member(X,L), !, zam(Y,L,Y1).
zam([X|Y],L,[X|Y1]):- zam(Y,L,Y1).
```

32. Дефинирайте предикат, който проверява дали N-тия елемент на списък е равен на N+1-вия.
```
equN(1,[X,X|Y]):- !.
equN(N,[X|Y]):- N1 is N-1, equN(N1,Y).
```

33. Дефинирайте предикат, който изтрива всички елементи на списъка L, които са на позиция кратна на 4.
```
dl(L,L1):- dls(L,L1,1).
dls([X|Y],Y1,N):- N mod 4 =:= 0, !, M is N+1, dls(Y,Y1,M).
dls([X|Y],[X|Y1],N):- M is N+1,dls(Y,Y1,M).
dls([],[],_).
II н.
d([X,Y,Z,_|T],[X,Y,Z|T1]):- d(T,T1),!.
d(L,L).
III н.
del4(L,L1):- p(L,L1,1).
p([],[],_):- !.
p([X|T],T1,4):- !, p(T,T1,1).
p([X|T],[X|T1],N):- N1 is N+1, p(T,T1,N1).
```

Упражнение 5 по Пролог (LPA Win-Prolog)

(29.11.2004 г.)

I. Динамични бази данни (ДБД) (**assert**(<клауза.>), **asserta**(<клауза.>), **assertz**(<клауза.>), **listing**(<име на предикат с/без брой арг.>), **retract**(<клауза>), **retractall**(<лява част на клауза>), **clause**(<лява част на клауза>,<дясна част на клауза>), **abolish**(<име на предикат>,<брой арг.>)) и предикат от по-висок ред

findall(<шаблон за елементите на списъка от решения>,<цел>,<списък от решения на целта>).

Задача 1 Дадени са факти, задаващи студентите и факти, задаващи оценките на студент от изпит по даден предмет. Дефинирайте предикат, който намира списъка от предметите, по които си е взел изпита даден студент (има оценка >2).

```
student(ivan).          student(maria).
student(petar).         student(dimo).
exam(ivan,m,2).         exam(ivan,e,6).
exam(maria,h,3).        exam(peter,e,2).
exam(ivan,m,4).         exam(maria,a,3).
exam(maria,h,4).        exam(maria,e,4).
exam(maria,h,6).        exam(maria,m,5).
% Предикат, който задава кога студентът S си е взел изпита X
take(S,X):-exam(S,X,M),M>2.
% Предикат, който задава кои изпити (като списък) си е взел студентът S
collect(S,L):-student(S), findall(X,take(S,X),L).
collect1(S,L):-student(S),findall(X,take(S,X),L1),sort(L1,L).
```

Задача 2

а) Напишете предикат за въвеждане на факти в ДБД, посочващи оценките на студенти по дадените с предиката subject предмети.

б) Напишете предикат, чрез който се намира средния успех на даден студент.

в) Напишете предикат, който записва в ДБД факти за средния успех на всеки студент.

г) Напишете предикат, който образува списък от всички студенти с техния среден успех.

```
% Факти за изучаваните предмети
subject('Логическо програмиране').
subject('Анализ').
subject('ЛААГ').
subject('Алгебра').
subject('Дескриптивна геометрия').
subject('Диференциални уравнения').
subject('Числени методи').
subject('Теория на вероятностите').
subject('Английски').
subject('Интегрирани среди и приложения').
subject('Компютърна архитектура').
% Попълване на динамична БД за оценките на студенти по дадените предмети
% За край на въвеждането - име exit
dialog:-repeat,write('Въведете име'),read(N),atom(N), (N=exit,!;subject(S),inp(S,M),assert(mark(N,S,M)),fail).
inp(S,M):-repeat,write('Въведи оценка по '),write(S),read(M),number(M),M>=2,M=<6,!.
% Намиране на средно аритметично на елементите на списък от числа (закръгляване до стотни)
avelist(L,A):- sumlist(L,S),length(L,N),A1 is (S*1000//N+5)//10,A is A1/100.
sumlist([],0).
sumlist([H|T],S):- sumlist(T,S1),S is S1+H.
% Намиране на среден успех на даден студент
ave(Name,A):-findall(X,mark(Name,_,X),L),avelist(L,A).
% Попълване на ДБД със средните оценки на всички студенти
 % I начин
fill:-mark(X,_,_),saveunique(X),fail.
saveunique(X):- clause(avmark(X,_),true),!.
saveunique(X):- ave(X,Y),assert(avmark(X,Y)).
 % II начин
fill1:-findall(X,mark(X,_,_),L),sort(L,L1),member(X,L1),ave(X,A),assert(avmark(X,A)),fail.
```

```
% Получаване на списък от студентите и техните средни оценки
    fill3(R):- findall(X,mark(X,_,_),L),sort(L,L1),findall(stud(X,Y),(member(X,L1),ave(X,Y)),R).
% Изтриване на ДБД
    clear:-retractall(mark(_,_,_)),retractall(avmark(_,_)).
```

Задача 3 Дадени са факти, задаващи кое дете от кой град е, като детето е стуктура с аргументи име и възраст. Дефинирайте предикат, който получава списък на всички деца, сортиран по възрастта им.

```
from(child(ivan,4),plovdiv).
from(child(stoian,5),sofia).
from(child(maria,4),sofia).
from(child(ivo,6),pleven).
split(_,[],[],[]).
split(X,[H|T],[H|T1],T2):-arg(2,H,Y),Y<X,!,split(X,T,T1,T2).
split(X,[H|T],T1,[H|T2]):-split(X,T,T1,T2).
sortq([],[]).
sortq([H|T],L):-arg(2,H,Y),split(Y,T,T1,T2),sortq(T1,L1),sortq(T2,L2),append(L1,[H|L2],L).
children(L):-findall(X,from(X,Y),L1),sortq(L1,L),!.
```

II. Обработка на структури – предикати **functor**(<терм>, <функтор>, <брой аргументи>) и **arg**(<номер на аргумент (от 1)>, <терм>, <(стойност на) аргумент с дадения номер>).

1. Предикат, чрез който се изтриват всички елементи на списък, които са съставни терми с първи аргумент a. Пример: [a,g(a,1,m),j(1,2),j(a),1,[2,4],s(a,d(l))] --> [a,j(1,2),1,[2,4]]

```
del([],[]).
del([H|T],T1):-arg(1,H,a),!,del(T,T1).
del([H|T],[H|T1]):-del(T,T1).
```

2. Напишете предикат, който в даден списък премахва всички елементи структури, имащи 1-ви и 3-ти аргумент едно и също число.
   Пример: [f(1,abc,1),f(1,abc),sf(7,7,7,7,7),[2],xx,18,d(w,w,w)] --> [f(1,abc),[2],xx,18,d(w,w,w)]

3. Предикат, чрез който се изтрива първият аргумент на даден съставен терм, ако той е равен на a.

Първо решение

```
    delarg(T,T1):- T=..[X,a|L],!,T1=..[X|L].
    delarg(T,T).
```

Второ решение

```
    delarg1(T,T1):-arg(1,T,a),!,functor(T,F,N),N1 is N-1,functor(T1,F,N1),copyarg(N1,T,T1).
    delarg1(T,T).
    copyarg(N,T,T1):- N>0,I is N+1,arg(I,T,X),arg(N,T1,X),N1 is N-1,copyarg(N1,T,T1).
    copyarg(0,_,_).
```

4. Дефинирайте предикат, който премахва последния аргумент на дадена структура, ако самият този аргумент не е списък, но е структура, имаща поне един аргумент.
   Примери: strct(a,f(2)) --> strct(a); f(h(1,2,3)) --> f; h(a,[w]) --> h(a,[w]).

5. Напишете предикат, който премахва първите N аргумента на дадена структура при дадено N.
   Примери при N=2: f(1,2,3) --> f(3); h(a,b) --> h; sc([2]) --> sc; x(a,b,c,d,e([2])) --> x(c,d,e([2])).

6. Предикат, чрез който във всички елементи на даден списък, които са съставни терми с поне два аргумента и вторият им аргумент е даден терм X, заменя този аргумент X с даден терм Y.
   Пример при X=a и Y=b: [d(m,a,1),2,d(k),[1,2],n(1,a,3),l] --> [d(m,b,1),2,d(k),[1,2],n(1,b,3),l]

```
subst(_,_,[],[]).
subst(X,Y,[H|T],[H1|T1]):- H=..[A,B,X|L],!,H1=..[A,B,Y|L],subst(X,Y,T,T1).
subst(X,Y,[H|T],[H|T1]):-subst(X,Y,T,T1).
```

7. Напишете предикат, който във всички елементи на даден списък, които са съставни терми с функтор равен на даден атом X и с първи аргумент атом Arg1, заменя функтора X с атома Arg1, а аргумента Arg1 заменя с даден терм Y.
   Пример при X=abc и Y=[w,f(9,a)]: [f(xyz,1),abc(xyz,7),[abc,yy,zz],abc(19,23)] -->
   [f(xyz,1),xyz([w,f(9,a)],7),[abc,yy,zz],abc(19,23)]