Brojac.pl

```
/* Broj atom u listi jednakih datom atomu */

brojac(_, [],0).
brojac(A,[B|R],N):-atom(B), A=B, !, brojac(A,R,N1), N is N1+1.
brojac(A,[B|R],N):-brojac(A,R,N).
```

Prolog zadaci za vežbu

1.1 Liste

1. Napisati predikat koji proverava da li je dati element član date liste.

```
sadrzi(G, [G|R]):-!.
sadrzi(X, [G|R]) :- sadrzi(X,R).
```

2. Napisati predikat kojim se odredjuje broj elemenata u listi L.

```
brojel([],0).
brojel([G|R],B):-brojel(R,B1),B is B1+1.
```

3. Napisati predikat kojim se određuje suma elemenata u listi brojeva L i predikat kojim se određuje aritmeticka sredina elemenata liste brojeva..

```
sumael([],0).
sumael([G|R],S):-sumael(R,S1),S is S1+G.

/* Definisi predikat kojim se odredjuje aritmeticka sredina elemenata liste brojeva. */
arsr(L,A):-brojel(L,K),sumael(L,S),A is S/K.
```

4. Napisati predikat koji za dati prirodan broj N učitava elemente liste duzine N.

```
unesi(0, []):-!.
unesi(N, [G|R]):-
        write('unesi element '),
        read(G), nl,
        M is N-1,
        unesi(M,R).
```

IZLAZ
2 ?- unesi(5,X).
unesi element 100.

unesi element 200.

unesi element 65.

unesi element 34.

unesi element -21.

X = [100, 200, 65, 34, -21].

5. Napisati predikat kojim se odredjuje broj parnih elemenata u listi celih brojeva.

```
brojparnih([],0).
brojparnih([G|R],K):-G mod 2 =:= 0,brojparnih(R,K1),K is K1+1.
brojparnih([G|R],K):-G mod 2 =\= 0,brojparnih(R,K).
```

6. Napisati predikat kojim se odredjuje suma pozitivnih elemenata u listi celih brojeva.

```
sumapoz([],0).
sumapoz([G|R],S):-G>0,sumapoz(R,S1),S is S1+G.
sumapoz([G|R],S):-G=<0,sumapoz(R,S).
```

7. Napisati predikat kojim se odredjuje N-ti clan liste.

```
nticlan([G|R],1,G).
nticlan([G|R],K,N):-K>1,K1 is K-1,nticlan(R,K1,N).
```

8. Napisati predikat kojim se od date liste brojeva formira nova koja sadrzi samo pozitivne brojeve.

```
samopoz([],[]).
samopoz([G|R],[G|R1]):-G>0,samopoz(R,R1).
samopoz([G|R],NL):-G=<0,samopoz(R,NL).
```

9. Napisati predikat koji briše prvo pojavljivanje datog elementa iz date liste i predikat koji briše sva pojavljivanja datog elementa iz date liste.

```
brisi1(X,[],[]):-!.
brisi1(G, [G|R], R) :- !.
brisi1(X, [G|R], [G|R1] ) :- brisi1(X, R, R1).

brisi(X,[],[]):-!.
```

brisi(G, [G|R], R1) :- brisi(G, R, R1),!.
brisi(X, [G|R], [G|R1]) :- brisi(X, R, R1).

%verzija 2

```
%brisi samo prvo pojavljivanje

brisi1(X, [], []).
brisi1(X, [X|R], R):- !.
brisi1(X, [G|R], [G|R1]):- G \== X,
                           brisi1(X, R, R1).

%brisanje elemenata liste
%brisi(X, L, LR) - X brise iz L da dobijemo LR

brisi(X, [], []).
brisi(X, [X|R], R1):- brisi(X, R, R1).
brisi(X, [G|R], [G|R1]):- G \== X,
                          brisi(X, R, R1).
```

10. Napisati predikat koji obrce datu listu.
obrni(L, Obrnuta) :- obrni(L, [], Obrnuta).

obrni([G|R], A, Y) :- obrni(R, [G|A], Y).
obrni([], A, A).

```
%obrtanje liste, verzija 2
%[1, 2, 3]  [3, 2, 1]
obrni(L, LR):- obrni(L, [], LR).
obrni([], L, L).
obrni([G|R], L, LR):- obrni(R, [G|L], LR).
```

11. Napisati predikat kojim se briše K -ti element liste.
izbaci([],K,[]).
izbaci([G|R],1,R).
izbaci([G|R],K,[G|R1]):-K>1,K1 is K-1, izbaci(R,K1,R1).

12. Napisati predikat kojim se odredjuje broj pojavljivanja elementa X u listi L.
brp([],X,0).
brp ([G|R],X,K):-G=:=X, brp (R,X,K1),K is K1+1.
brp ([G|R],X,K):-G=\=X, brp (R,X,K).

13. Napisati predikat kojim se od date liste formira nova tako sto se svaki negativan element duplira, tj. dva puta upisuje u novu listu.
doubleit([],[]).

```
doubleit([G|R],[G,G|R1]):- G<0,doubleit(R,R1).
doubleit([G|R],[G|R1]):- G>=0,doubleit(R,R1).
```

14. Napisati predikat kojim od date liste formira nova dobijena izbacivanjem K-tog elementa.

```
/* L=[1,4,2,9,-3]  K=4   NL=[1,4,2,-3] */
izbaci([],K,[]).
izbaci([G|R],1,R).
izbaci([G|R],K,[G|R1]):-K>1,K1 is K-1, izbaci(R,K1,R1).
```

15. Napisati predikat koji dodaje dati element na pocetak date liste i predikat koji dodaje dati element na kraj date liste.

```
dodaj_poc(X, L, [X|L]).

%dodaj element na kraj liste
%[X, Y|R] X=1 Y = 2 R=[3, 4]
dodaj_kraj(X, [], [X]).
dodaj_kraj(X, [G|R], [G|R1]) :- dodaj_kraj(X, R, R1).
```

16. Napisati predikat koji ispisuje poslednja tri elementa date liste i predikat koji ispisuje da li su data tri elementa uzastopni clanovi date liste.

```
poslednja3(X,Y,Z, [X,Y,Z]):-!.
poslednja3(X,Y,Z, [_|R]) :- poslednja3(X,Y,Z,R).

uzastopni(X,Y,Z, [X,Y,Z|_]).
uzastopni(X,Y,Z, [_|R]):-uzastopni(X,Y,Z,R).
```

17. Napisati predikat koji brise prvi clan liste i predikat koji brise poslednji clan liste.

```
brisi_prvi([G|R], R).

%brisi_poslednji([], _):-!, fail.
brisi_poslednji([_],[]):-!.
brisi_poslednji([G|R], [G|R1]) :- brisi_poslednji(R,R1).
```

18. Napisati predikat koji za dato N vraca prvih N elemenata date liste.

```
prvih_n(L,N,_):-duzina(L,M), N>M, write('prekratka lista'), !, fail.
prvih_n(L, 0, []):-!.
prvih_n([G|R], N, [G|R1]) :- M is N-1,
                             prvih_n(R,M, R1).

duzina([],0)                                .
```

```
duzina([G|R], X) :- duzina(R, Y), X is Y+1.
```

19. %spajanje dve liste
%spoji(L1, L2, L)

```
spoji([], L, L).
spoji([G|R], L, [G|LR]):- spoji(R, L, LR).
```

20. %zameniti dva elementa liste
%zameni(X, Y, L, LR)
%zameni X sa Y u L da dobijemo LR
%X = 2, Y= 3 [1, 2, 2, 4] [1, 3, 3, 4]

```
zameni(X, Y, [X|R], [Y|R1]):- zameni(X, Y, R, R1).
zameni(X, Y, [G|R], [G|R1]):- G \== X,
                              zameni(X, Y, R, R1).
zameni(X, Y, [], []).
```

21. Napisati predikat koji datu listu deli na listu parnih elemenata i listu neparnih elemenata.

```
podeli([],[],[]).
podeli([G|R], [G|R1], R2) :- uslov(G), podeli(R,R1,R2),!.
podeli([G|R], R1, [G|R2]) :- podeli(R,R1,R2).

uslov(X) :- X mod 2 =:= 0.
```

22. Napisati predikat koji datu listu deli na dve liste, listu pozitivnih elemenata i listu negativnih elemenata.

```
%podeliti listu na dve liste - listu pozitivnih
%i listu negativnih elemenata
%podeli(L, L1, L2) - L1 lista pozitivnih
%L2 lista negativnih.

podeli([], [], []).
podeli([G|R], [G|R1], LN):- G >= 0,
                            podeli(R, R1, LN).
podeli([G|R], LP, [G|R1]):- G < 0,
                            podeli(R, LP, R1).
```

23. %podeliti listu na sve moguce nacine

```
%[1, 2, 3]
%[] [1, 2, 3]
%[1] [2, 3]
%[1, 2] [3]
% itd....
%podeli(L, L1, L2).
```

```
podeli1([], [], []).
podeli1([G|R], [G|R1], Y):- podeli1(R, R1, Y).
podeli1([G|R], X, [G|R1]):- podeli1(R, X, R1).
```

24. Napisati predikat koji za dato N ispisuje prvih N redova Paskalovog trougla.

```
pascal(N):- redovi(0,N).

redovi(A,B) :- A>B,!.
redovi(A,B) :- red(A), nl,
               A1 is A+1,
               redovi(A1,B).

red(B):-koef(0,B).

koef(A,B):-A>B, !.
koef(A,B):-koeficijent(A,B,X),
           write(X),write(' '),
           A1 is A+1,
           koef(A1,B).

%koeficijent(0,0,1):-!.
koeficijent(0,B,1):-!.
koeficijent(B,B,1):-!.

koeficijent(A,B,X):- A1 is A-1, B1 is B-1,
                     koeficijent(A,B1,Y),
                     koeficijent(A1,B1,Z),
                     X is Y+Z.
```

25. Napisati predikat koji za datu listu cifara vraca broj odredjen tim ciframa.

```
izdvoj_poslednji([G], G, []):-!.
izdvoj_poslednji([G|R],X,[G|R1]):-izdvoj_poslednji(R,X,R1).

pretvori([], 0):-!.
pretvori(L, X):-izdvoj_poslednji(L, Poslednji, Ostatak),
                pretvori(Ostatak, Y),
                X is Poslednji+10*Y.
```

26. %maksimalni element liste
%maxL(L, X)

```
maxL([X], X).
maxL([G|R], X):- maxL(R, Y),
                 G < Y,
                 X is Y.
maxL([G|R], X):- maxL(R, Y),
                 G >= Y,
                 X is G.


maxL1([X], X).
maxL1([G|R], Y):- maxL1(R, Y),
                  G < Y.
```

```
maxL1([G|R], G):- maxL1(R, Y),
                  G >= Y.
```

27. Napisati predikat koji za dato N vraca sve kombinacije cifara 0 i 1 duzine N.

```
svi01(X) :- generisi(L, X),nl, write(L), fail.

generisi([],0).
generisi([0|R], N):-N>0, N1 is N-1, generisi(R,N1).
generisi([1|R], N):-N>0, N1 is N-1, generisi(R,N1).
```

28. Neka su u bazi znanja date reci ormar,torba,marko,banja,sto. Pod mutacijom reci X i reci Y pri cemu je neki sufiks reci X prefiks reci Y podrazumeva se rec dobijena od na sledeci nacin: npr mutacija za reci torba i banja je tornja. Napisati predikat koji generise sve takve mutacije.

```
rec(ormar).
rec(torba).
rec(marko).
rec(banja).
rec(sto).

mutacija :- mutiraj(A), name(X,A), write(X),nl, fail.
mutiraj(A) :- rec(Prva), rec(Druga),
              Prva\==Druga,
              name(Prva,X), name(Druga, Y),
              dodati(X1,X2,X), X2\==[],
              dodati(X2,Y2,Y),
              dodati(X1,Y2,A).

dodati([],L,L).
dodati([G|R], L2, [G|R2]):-dodati(R,L2,R2).
```

29. Cetiri coveka se zovu Pera, Mika, Laza i Jova, a prezivaju Peric, Mikic, Lazic i Jovic. Oni imaju cetiri sina koji se takodje zovu Pera, Mika, Laza i Jova. Pretpostavimo sledece:
(a) Niko od oceva se ne zove u skladu sa svojim prezimenom.
(b) Niko od sinova se ne zove u skladu sa svojim prezimenom.
(c) Niko od sinova se ne zove isto kao i otac.
(d) Peric stariji se zove isto kao Mikin sin.
(e) Lazin sin se zove Pera.
Napisati predikat koji odredjuje imena oceva i sinova.

```
ime(pera).
ime(laza).
ime(jova).
ime(mika).

prezime(peric).
prezime(lazic).
prezime(jovic).
prezime(mikic).
```

```
kandidat(otac_sin(Prezime,Otac,Sin)):- prezime(Prezime),
                                       ime(Otac),
                                       ime(Sin).

kandidat([]).
kandidat([G|R]):-kandidat(G), kandidat(R).

u_skladu(pera,peric).
u_skladu(jova,jovic).
u_skladu(mika,mikic).
u_skladu(laza,lazic).

prezimena_u_neskladu([]).
prezimena_u_neskladu([otac_sin(Prezime,Otac,Sin)|R]) :-
not(u_skladu(Otac,Prezime)), not(u_skladu(Sin,Prezime)),
prezimena_u_neskladu(R).

razlicita_imena_oceva([]).
razlicita_imena_oceva([otac_sin(_,Ime,_)|R]):-
not(clan(otac_sin(_,Ime,_), R)),
razlicita_imena_oceva(R).

razlicita_imena_sinova([]).
razlicita_imena_sinova([otac_sin(_,_,Ime)|R]):-
not(clan(otac_sin(_,_,Ime), R)),
razlicita_imena_sinova(R).

sinovi_razlicito_od_oceva([]).
sinovi_razlicito_od_oceva([otac_sin(Prezime,Ime,Ime)|R]):-!,fail.
sinovi_razlicito_od_oceva([otac_sin(_,_,_)|R]):-sinovi_razlicito_od_oceva(R).

peric_otac_mikin_sin(L) :- clan(otac_sin(peric,Ime,_),L),
clan(otac_sin(_,mika,Ime),L).

lazin_sin_pera(L) :- clan(otac_sin(_,laza,pera),L).

clan(G, [G|R]):-!.
clan(X, [G|R]):-clan(X,R).


uslovi(L):-
kandidat(L),
razlicita_imena_oceva(L),
razlicita_imena_sinova(L),
prezimena_u_neskladu(L),
sinovi_razlicito_od_oceva(L),
peric_otac_mikin_sin(L),
lazin_sin_pera(L).

ispisi_resenja:-
uslovi([otac_sin(peric, OtacPeric,SinPeric),
        otac_sin(lazic, OtacLazic,SinLazic),
        otac_sin(jovic, OtacJovic,SinJovic),
        otac_sin(mikic, OtacMikic,SinMikic)]),
write('Peric: '), write(OtacPeric), write(' '), write(SinPeric),nl,
write('Lazic: '), write(OtacLazic), write(' '), write(SinLazic),nl,
write('Jovic: '), write(OtacJovic), write(' '), write(SinJovic),nl,
```

```
write('Mikic: '), write(OtacMikic), write(' '), write(SinMikic),nl,
fail.
```

%verzija 2 resenja

```
%ime oca, ime sina, prezime
resi(L):- L= [[laza, pera, _], [X, _, peric], [mika, X, _], _],
             sadrzi([jova, _, _], L),
             sadrzi([pera, _, _], L),
             sadrzi([_,laza, _], L),
             sadrzi([_, jova, _], L),
             sadrzi([_, mika, _], L),
             sadrzi([_, _, mikic], L),
             sadrzi([_, _, jovic], L),
             sadrzi([_, _, lazic], L),
             \+(sadrzi([mika, _, mikic], L)),
             \+(sadrzi([pera, _, peric], L)),
             \+(sadrzi([jova, _, jovic], L)),
             \+(sadrzi([laza, _, lazic], L)),
             \+(sadrzi([_, mika, mikic], L)),
             \+(sadrzi([_, pera, peric], L)),
             \+(sadrzi([_, jova, jovic], L)),
             \+(sadrzi([_, laza, lazic], L)),
             \+(sadrzi([mika, mika, _], L)),
             \+(sadrzi([pera, pera, _], L)),
             \+(sadrzi([jova, jova, _ ], L)),
             \+(sadrzi([laza, laza, _], L)).

/* sadrzi(X, [X|_]).
sadrzi(X, [G|R]):- sadrzi(X, R).
*/
```

30. Napisati predikat :
a) umetni(X,L,NL) kojim se umece broj X u neopadajucu listu L i dobija neopadajuca lista NL
b) sort2(L,NL) kojim se formira neopadajuca lista NL od liste L

```
umetni(X, [], [X]).
umetni(X, [G|R], [G|R1]):-X>G, umetni(X,R,R1).
umetni(X, [G|R], [X,G|R]):-X=<G.

sort2([],[]).
sort2([G|R], NL):- sort2(R,R1), umetni(G, R1, NL).
```

31. Napisati predikat kojim se za dati prirodan broj N formira lista
a) [N, N-1, ..., 1]
b) [1, 2, 3, ..., N]

```
pa(0,[]).
pa(N,[N|R]):-N>0, N1 is N-1, pa(N1,R).
```

```
pb(N,L):-pb(N,L,1).
pb(N,[N],N).
pb(N,[I|R],I):-I<N, I1 is I+1, pb(N,R,I1).
```

32. Napisati predikat kojim se formira lista od prirodnih brojeva deljivih sa 5 i manjih od datog broja N. N=30 L=[5,10,15,20,25]

```
form(N,L):-form(N,L,5).
form(N,[],I):-I>=N.
form(N,[I|R],I):-I<N, I1 is I+5, form(N,R,I1).
```

33. Napisati predikat kojim se formira lista od prvih N prirodnih brojeva deljivih sa 5. Npr za N=3 L=[5,10,15]

```
form(N,L):-form(N,L,5).
form(0,[],I).
form(N,[I|R],I):-N>0, N1 is N-1, I1 is I+5, form(N1,R,I1).
```

34. Spojiti dve sortirane liste u trecu tako da ona bude sortirana.

```
spoji2([],L,L).
spoji2(L,[],L).
spoji2([G1|R1],[G2|R2],[G1|R]):-G1<G2,spoji2(R1,[G2|R2],R).
spoji2([G1|R1],[G2|R2],[G2|R]):-G1>=G2,spoji2([G1|R1],R2,R).
```

35. Napisati predikat kojim se u listi L duplira K-ti element.
Primer: L=[1,4,2,9,-3] K=4 NL=[1,4,2,9,9,-3]

```
dupliraj([],K,[]).
dupliraj([G|R],1,[G,G|R]).
dupliraj([G|R],K,[G|R1]):-K>1,K1 is K-1, dupliraj(R,K1,R1).
```

1.2 Vežbanje pred pismeni

1. Neka je data baza znanja u Prologu cinjenicama:
film(Naziv_filma, Zanr_filma, Ime_reditelja, Sifra_glumca)
glumac(Sifra_glumca, Ime_glumca, God_rodj, Mesto_rodj)

a) Napisati pravilo filmski_umetnik(X) X je filmski_umetnik ako je X reditelj nekog filma i X igra u nekom filmu.

```
filmski_umetnik(X):-film(_,_,X,_),glumac(SX,X,_,_),film(_,_,_,SX).
```

b) Napisati pravilo glumac_2(X) X igra u bar dva razlicita filma

```
glumac_2(X):-glumac(SX,X,_,_),film(F1,_,_,SX), film(F2,_,_,SX), not(F1=F2).
```

c) Napisati pravilo opsti_glumac(X) X igra u bar dva filma razlicitog zanra

opsti_glumac(X):-glumac(SX,X,_,_),film(_,Z1,_,SX),film(_,Z2,_,SX), not(Z1=Z2).

d) Napisati pravilo zanrovski_glumac(X,Y) glumac cije je ime X igra u filmu zanra Y.

zanrovski_glumac(X,Y):-glumac(SX,X,_,_),film(_,Y,_,SX).

2. Definisati predikat kojim se određuje maksimum
a) za dva broja A i B.
b) za tri broja A, B i C.

max(A,B,A):-A>B.
max(A,B,B):-A=<B.
max(A,B,C,X):-max(A,B,X1),max(X1,C,X).

3. Definisati predikat kojim se za dati prirodan broj N određuje
a) broj cifara
b) suma cifara
c) broj neparnih cifara
d) maksimalna cifra
e) K-ta cifra (gledano s desna u levo)

```
bc(N,1):-N<10.
bc(N,K):-N>=10,N1 is N//10,bc(N1,K1),K is K1+1.

sc(0,0).
sc(N,K):-N>0,N1 is N//10,sc(N1,K1),K is K1+ N mod 10.

bnc(N,0):- N<10,N mod 2=:=0.
bnc(N,1):- N<10,N mod 2=:=1.
bnc(N,K):-N>=10,N mod 2=:=1,N1 is N//10,bnc(N1,K1),K is K1+1.
bnc(N,K):-N>=10,N mod 2=:=0,N1 is N//10,bnc(N1,K).

mc(N,N):-N<10.
mc(N,K):-N>=10,N1 is N//10,mc(N1,K1),N2 is N mod 10,max2(N2,K1,K).

kc(N,1,X):-X is N mod 10.
kc(N,K,X):-K>1,K1 is K-1, N1 is N//10,kc(N1,K1,X).
```

4. Napisati predikat koji za dati ceo broj ispisuje rečima njegove cifre.

```
cifre(0,nula).
cifre(1,jedan).
```

```
cifre(2,dva).
cifre(3,tri).
cifre(4,cetiri).
cifre(5,pet).
cifre(6,sest).
cifre(7,sedam).
cifre(8,osam).
cifre(9,devet).

ispisi(Broj) :- Broj>=0, Broj=<9,
            cifre(Broj, X),
            write(X), nl, !.

ispisi(Broj) :-
            B1 is Broj // 10,
            ispisi(B1),
            B2 is Broj mod 10,
            cifre(B2,X),
            write(X),nl
            .

start :- write('unesi prirodan broj \'),
        read(X),
        provera(X),
        ispisi(X).

provera(X):- X>=0,!.
provera(X):- write('broj mora biti prirodan!'), nl, fail.
```

5. Za datu bazu znanja koje predstavlja porodicno stablo, napisati predikate otac, majka, brat, ujak, predak.

```
musko(mihajlo).
musko(stevan).
musko(rajko).
musko(mladen).
musko(petar).

zensko(milena).
zensko(milica).
zensko(jelena).
zensko(mina).
zensko(senka).
zensko(maja).
```

roditelj(mihajlo, milica).
roditelj(mihajlo, rajko).
roditelj(mihajlo, senka).
roditelj(milena, milica).
roditelj(milena, rajko).
roditelj(milena, senka).
roditelj(stevan, mladen).
roditelj(stevan, jelena).
roditelj(milica, mladen).
roditelj(milica, jelena).
roditelj(rajko, petar).
roditelj(rajko, mina).
roditelj(maja, petar).
roditelj(maja, mina).

```
otac(X, Y):- musko(X), roditelj(X, Y).
majka(X, Y):- zensko(X), roditelj(X, Y).
```

brat(X, Y):- musko(X), roditelj(Z, X),
roditelj(Z, Y),
X\==Y.

```
brat2(X,Y):-musko(X),roditelj(Z,X), roditelj(Z,Y), not(X=Y).

brat3(X,Y):-musko(X),otac(Z,X), otac(Z,Y),
                    majka(M,X), majka(M,Y), not(X=Y).
```

ujak(X, Y) :- brat(X, Z), majka(Z, Y).

predak(X, Y):- roditelj(X,Y).
predak(X, Y):- roditelj(X, Z), predak(Z,Y).

```
sestra(X, Y):- zensko(X), roditelj(Z, X), roditelj(Z, Y), X\==Y.

tetka(X, Y):- roditelj(Z,Y), sestra(X, Z).

sestraodujaka(X, Y):- ujak(Z, Y), roditelj(Z, X), zensko(X).
```

6. suma cifara datog broja

```
suma2(N, N):- N < 10.
suma2(N, S):- N >= 10,
              M is N // 10,
                  suma2(M, S1),
                  S is S1 + (N mod 10).

% verzija 2
```

sumac(0,0).
sumac(N,K):-N>0,N1 is N//10,sumac(N1,K1),K is K1+ N mod 10.

```
7. %obrni cifre broja
obrni(N, N, P):- N < 10,
                 P is 10.
obrni(N, R, P):- N >= 10,
                 N1 is N // 10,
                 obrni(N1, R1, P1),
                 R is (R1 + (N mod 10)*P1),
                 P is (P1*10).
```

8. Napisati predikat kojim se za prirodan broj N i datu duzinu D (D>0) određuju podbrojevi broja N dužine D (podbroj čine uzastopne cifre). Npr. N=51478 D=2 78, 47, 14, 51

```
podbroj(N,D,X):-stepen(10,D,S), p(N,S,X).
p(N,S,X):- N>=S/10,X is N mod S.
p(N,S,X):-N>=S,N1 is N//10, p(N1,S,X).
```

6. Napiši predikat kojim se određuje proizvod prirodnih brojeva od M do N.

```
proizvod(M,M,M).
proizvod(M,N,F):-N>M, N1 is N-1, proizvod(M,N1, F1), F is F1*N.
```

7. Definiši predikat kojim se od datog prirodnog broja N formira broj M dobijen izbacivanjem svakog pojavljivanja cifre C.

```
nbroj(0,0,C).
nbroj(N,M,C):-N>0, N mod 10=\=C, N1 is N//10,
                nbroj(N1,M1,C), M is M1*10+ N mod 10.
nbroj(N,M,C):-N>0, N mod 10=:=C, N1 is N//10, nbroj(N1,M,C).
```

8. Date su cinjenice roditelj(X,Y) i godina_rodjenja(X,G).

a) Napisati pravilo naslednik(X,Y)osoba X je naslednik osobe Y

```
naslednik(X,Y):-roditelj(Y,X).
naslednik(X,Y):-roditelj(Z,X),naslednik(Z,Y).
```

b) Napisati pravilo bar_dva(X,Y,Z)osoba X ima dva naslednika Y i Z rodjena iste godine.

```
bar_dva(X,Y,Z):-naslednik(Y,X), naslednik(Z,X), not(Y=Z),
                godina_rodjenja(Y,G),godina_rodjenja(Z,G).
```

c) Napisati pravilo predak_c(X,Y,G)osoba Y je predak osobe X rodjen godine G.

```
predak_c(X,Y,G):-naslednik(X,Y),godina_rodjenja(Y,G).
```

9. Ucenici nekog odeljenja nalaze se u koloni po jedan po visinama, u rastucem poretku. Date su cinjenice

pored_d(X,Y) - desno pored osobe X u koloni je osoba Y

godina(X,Y) - osoba X rodjena je godine Y

Napisati pravilo

a) pa(X,Y) - X je osoba koja je niza od osobe Y

b) pb(X,Y) - X je osoba koja je niza od osobe Y a rodjene su iste godine

c) pc(X,Y,Z) - osobe Y i Z su dve razlicite osobe koje su nize od osobe X

```
pa(X,Y):-pored_d(X,Y).
pa(X,Y):-pored_d(X,Z),pa(Z,Y).
pb(X,Y):-pa(X,Y),godina_rodjenja(X,G), godina_rodjenja(Y,G).
pc(X,Y,Z):-pa(Y,X),pa(Z,X),not(Y=Z).
```

10. Date su cinjenice

brzi(SX,SY) - automobil sifre SX brzi je od automobila SY

auto(Naziv_automobila, Sifra_automobila)

vlasnik(Ime_vlasnika, Naziv_automobila)

Napisati pravilo

a) p4a(X,Y) automobil naziva X je brzi od automobila naziva Y

b) p4b(X) lice cije je ime X ima automobil

c) p4c(X,Y) X je vlasnik brzeg automobila nego sto je Y

```
p4a(X,Y):-auto(X,SX), auto(Y,SY), brzi(SX,Sy).
p4b(X):-vlasnik(X,_).
p4c(X,Y):-vlasnik(X,A1), vlasnik(Y,A2), p4a(A1,A2).
```

11. Date su cinjenice, koje nam govore za svaka dva susedna cina u vojsci koji je visi.

visi_cin(zastavnik,vodnik).
visi_cin(major,zastavnik).
visi_cin(pukovnik,major).
visi_cin(generalmajor,pukovnik).
visi_cin(generalpukovnik,generalmajor).
visi_cin(general,generalpukovnik).

Napisati pravilo blizi(X,Y) u znacenju cin X je blizi generalskom cinu od cina Y.

```
blizi(X,Y):-visi_cin(X,Y).
blizi(X,Y):-visi_cin(X,Z),blizi(X,Y).
```

12. **provera da li je broj prost**

```
prost(X):- X1 is X // 2, prost_pom(X, X1).

prost_pom(X, 1):- true.
prost_pom(X, B):- B > 0,
                  X mod B =\= 0,
                  B1 is B - 1,
                  prost_pom(X, B1).
```

% verzija 2

```
prost(N):-prost(N,2).
prost(N,I):-I>N//2.
prost(N,I):-I=<N//2, N mod I=\=0, I1 is I+1, prost(N,I1).
```

13. Apsolutna vrednost – lose i dobre implementacije

```
%apsolutna vrednost
%losa implementacija
aps1(X, X):- X >= 0.
aps1(X, Y):- Y is -X.

%dobre implementacija
aps2(X, X):- X >= 0.
aps2(X, Y):- X < 0,
             Y is -X.

aps3(X, X):- X >= 0, !. %koriscenje operatora "cut"
aps3(X, Y):- Y is -X.
```

14. N-ti clan Fibonacijevog niza, verzija bez i sa operatorom !

```
fibonaci(1, 1).
fibonaci(2, 1).
fibonaci(N, R):- N > 2,
                 N1 is N - 1,
                 N2 is N - 2,
                 fibonaci(N1, R1),
                 fibonaci(N2, R2),
                 R is R1 + R2.

%N-ti clan Fibonacijevog niza, verzija II
fibonaci2(1, 1):- !.
fibonaci2(2, 1):- !.
fibonaci2(N, R):- N1 is N - 1,
                  N2 is N - 2,
                  fibonaci2(N1, R1),
                  fibonaci2(N2, R2),
                  R is R1 + R2.
```

15. izracunati stepen broja

```
stepen(X,0,1).
stepen(X,Y,R):- Y>0,
                                    Y1 is Y-1,
                                    stepen(X,Y1,R1),
                                    R is R1*X.
```

%verzija 2: definiši predikat kojim se određuje X^N ako je X realan a N ceo broj veći ili jednak 0.

```
stepen(_,0,1).
stepen(X,N,S):-N>0, N1 is N-1, stepen(X,N1,S1), S is S1*X.
```

16.Prema Goldbahovoj hipotezi, svaki paran broj moze se napisati kao zbir dva prosta broja. Napisati PROLOG predikat koji za dati paran broj X odreduje njegove Goldbahove sabirke.

```
gol_pom(X1, Y1, X1, Y1):- prost(X1),
                          prost(Y1),
                          !.
gol_pom(X, Y, X1, Y1):- X2 is (X1 + 1),
                            Y2 is (Y1 - 1),
                            gol_pom(X, Y, X2, Y2).
gol(N, X, Y):- Y1 is (N-2),
               gol_pom(X, Y, 2, Y1).
```

17. Definiši predikat kojim se određuje maksimalna cifra prirodnog broja N.

```
max2(A,B,A):-A>B.
max2(A,B,B):-A=<B.
mc(N,N):-N<10.
mc(N,K):-N>=10,N1 is N//10,mc(N1,K1),N2 is N mod 10,max2(N2,K1,K).
```

18. Definiši predikat kojim se određuje inverzan zapis prirodnog broja N

```
inv(N,X):-inv(N,X,0).
inv(0,X,X).
inv(N,X,Tek):-N>0,Tek1 is Tek*10 + N mod 10, N1 is N//10, inv(N1,X,Tek1).
```

19. Definiši predikat kojim se određuje suma delilaca prirodnog broja N, ne uključujući broj N.

```
sumad(N,S):-sumad(N,S,1).
sumad(N,0,I):-I>N//2.
sumad(N,S,I):-I=<N//2, N mod I=:=0, I1 is I+1, sumad(N,S1,I1), S is S1+I.
sumad(N,S,I):-I=<N//2, N mod I=\=0, I1 is I+1, sumad(N,S,I1).
```

20. % pairsums(L1, L2) koji za zadatu listu L1 vraca listu L2
% suma elemenata liste L1 na susednim pozicijama. Na primer,
% ako je L1 [1,3,6,10], onda L2 treba da bude [4,9,16].
% Ako L1 sadrzi manje od dva elementa, L2 treba postaviti na
% praznu listu.

```
pairsums([X, Y|R], [G|R1]):- G is X + Y, pairsums([Y|R], R1).
pairsums([X], []).
pairsums([], []).
```

21. Napisati predikat kojim se iz prirodnog broja N izbacuje K-ta cifra gledano s desna na levo.

```
%izbaci(N,K,X)
izbaci(N,1,X):-X is N//10.
izbaci(N,K,X):-K>1, K1 is K-1, N1 is N//10,
               izbaci(N1,K1,X1), X is X1*10+N mod 10.
```

22. Napisati predikat kojim se iz prirodnog broja N izbacuju sve cifre manje od 5.

```
izbacim5(0,0).
izbacim5(N,X):- N>0, N mod 10>=5, N1 is N//10,
                izbacim5(N1,X1), X is X1*10+N mod 10.
izbacim5(N,X):- N>0, N mod 10<5, N1 is N//10,
                izbacim5(N1,X).
```

23. Napisati predikat kojim se iz prirodnog broja N zamenjuje svako pojavljivanje cifre X sa cifrom Y.

```
zameni(0,_,_,0).
zameni(N,X,Y,A):- N>0, N mod 10=:=X, N1 is N//10,
                  zameni(N1,X,Y,A1), A is A1*10+Y.
zameni(N,X,Y,A):- N>0, N mod 10=\=X, N1 is N//10,
                  zameni(N1,X,Y,A1), A is A1*10+N mod 10.
```

24. Napisati predikat kojim se prikazuju (write(X)) prvih N prostih brojeva.

```
prost(N):-prost(N,2).
prost(N,K):-K>N//2.
prost(N,K):-K=<N//2,N mod K=\=0, K1 is K+1, prost(N,K1).

pisiProste(N):-pisiProste(N,2).
pisiProste(0,_).
pisiProste(N,K):-N>0, prost(K), write(K),nl, K1 is K+1, N1 is N-1, pisiProste(N1,K1).
pisiProste(N,K):-N>0, not(prost(K)),K1 is K+1, pisiProste(N,K1).
```

25. Napisati predikat kojim se prikazuje rastavljanje prirodnog broja N na proste faktore (write(X))

```
12  = 2 *2* 3

faktori(N):-faktori(N,2).
faktori(1,_).
faktori(N,K):-N>1, N mod K=:=0, write(K), nl, N1 is N//K, faktori(N1,K).
faktori(N,K):-N>1, N mod K=\=0, K1 is K+1, faktori(N,K1).
```

26. Odrediti sumu delioca prirodnog broja N, u sumu ne uključiti broj N.

```
sumaD(N,S):-sumaD(N,S,1).
sumaD(N,0,K):-K>N//2.
sumaD(N,S,K):-K=<N//2, N mod K=:=0, K1 is K+1, sumaD(N,S1,K1), S is S1+K.
sumaD(N,S,K):-K=<N//2, N mod K=\=0, K1 is K+1, sumaD(N,S,K1).
```

27. Proveriti da li je prirodan broj savrsen. Broj je savršen ako je jednak sumi svojih delioca isključujući njega samog. Npr 6=1+2+3 28=1+2+4+7+14

```
savrsen(N):-sumaD(N,N).
```

28. Ispisati sve savršene prirodne brojeve iz segmenta [a,b].

```
pisiSavrsene(A,B):-A>B.
pisiSavrsene(A,B):-A=<B, savrsen(A), write(A), nl, A1 is A+1, pisiSavrsene(A1,B).
pisiSavrsene(A,B):-A=<B, not(savrsen(A)), A1 is A+1, pisiSavrsene(A1,B).
```

1.3 Razno

1. Ko je udario macku Tunu? Znamo da:
macka je zivotinja
vlasnik psa voli zivotinje
ne bi udario zivotinju ko voli zivotinje
macku bi udario onaj koji bi je mozda udario i nije da je ne bi udario.

```
macka(tuna).
vlasnikpsa(janko).
mozda_udario(marko,tuna).
mozda_udario(janko,tuna).

zivotinja(X):-macka(X).
volizivotinje(X):-vlasnikpsa(X).
ne_bi_udario(X,Y):-volizivotinje(X),zivotinja(Y).
udario(X,Y):-mozda_udario(X,Y), \+(ne_bi_udario(X,Y)).
```

2. Data je baza znanja.

```
film(sutjeska, ratni, veljko_bulajic, s1).
film(sutjeska, ratni, veljko_bulajic, s2).
film(sutjeska, ratni, veljko_bulajic, s3).
film(walter_brani_sarajevo, ratni, hajrudin_krvavac,s1).
film(dr, komedija, soja_jovanovic,s1).
glumac( s1, bata_zivojinovic, 1927, beograd).
glumac( s2, ricard_barton, 1921, london).
glumac( s3, veljko_bulajic, 1924, sarajevo).
```

Ispisi filmske umetnike

```
filmski_umetnik(X):-film(_,_,X,_),glumac(SX,X,_,_),film(_,_,_,SX).
glumac_2(X):-glumac(SX,X,_,_),film(F1,_,_,SX), film(F2,_,_,SX), not(F1=F2).
opsti_glumac(X):-glumac(SX,X,_,_),film(_,Z1,_,SX),film(_,Z2,_,SX),
not(Z1=Z2).
%film( Naziv_filma, Zanr_filma, Ime_reditelja, Sifra_glumca)
%glumac( Sifra_glumca, Ime_glumca, God_rodj, Mesto_rodj)
```

3. Hanojeve kule

```
hanoj(1, X, _, Z):- write('prebaci sa '),
                    write(X),
                    write(' na '),
                    write(Z),
                    nl.
```

```
hanoj(N, X, Y, Z):- N > 1,
                    M is N - 1,
                    hanoj(M, X, Z, Y),
                    hanoj(1, X, _, Z),
                    hanoj(M, Y, X, Z).
```

4. bojenje grafa

```
bojenje(Srb, Cg, Mak, Hrv, Slo, Bih, Madj, Bug, Rum):-
sused(Srb, Cg),
sused(Srb, Mak),
sused(Srb, Hrv),
sused(Srb, Bih),
sused(Srb, Madj),
sused(Srb, Bug),
sused(Srb, Rum),
sused(Cg, Hrv),
sused(Cg, Bih),
sused(Hrv, Slo),
sused(Hrv, Bih),
sused(Hrv, Madj),
sused(Madj, Rum),
sused(Rum, Bug).

boja(zuta).
boja(plava).
boja(crvena).

sused(X,Y):-boja(X), boja(Y), X\==Y.
```

5. Ko laze taj krade.

```
Ko krade i uhvacen je u kradi taj ide u zatvor.
Al Kapone laze.
Al Kapone je uhvacen u kradi.
Laki Luciano laze.
Napisati PROLOG program koji opisuje navedene cinjenice i pravila. Koje
odgovore PROLOG daje na upite
  a) da li Al Kapone ide u zatvor
  b) da li Laki Luciano ide u zatvor?

krade(X):- laze(X).
zatvor(X):- krade(X), uhvacen_u_kradji(X).

laze(alKapone).
uhvacen_u_kradji(alKapone).
laze(lakiLuciano).
```

6. Pomocni zadatak

Definiši predikat kojim se proverava da li su dva elementa

a. jedan pored drugog
b. prvi element desno od drugog

```
pored(X,Y,[X,Y|_]).
pored(X,Y,[Y,X|_]).
```

```
pored(X,Y,[_|R]):-pored(X,Y,R).

desno(X,Y,[Y,X|_]).
desno(X,Y,[_|R]):-desno(X,Y,R).
```

Primetimo da pomoću predikata pored (desno) možemo i da generišemo sve liste u kojima su elementi jedan pored drugog (tj. desno)

```
?- pored(1,3,L).
L = [1, 3|_G201] ;
L = [3, 1|_G201] ;
L = [_G197, 1, 3|_G204]

?- L=[_,_,_],pored(1,3,L).
L = [1, 3, _G353] ;
L = [3, 1, _G353] ;
L = [_G347, 1, 3] ;
L = [_G347, 3, 1] .

?- L=[_,_,_],pored(1,3,L),clan(5,L).
L = [1, 3, 5] ;
L = [3, 1, 5] ;
L = [5, 1, 3] ;
L = [5, 3, 1] ;
false.
```

7.

Postoji pet kuća, svaka različite boje u kojoj žive ljudi različitih nacionalnosti sa različitim kućnim ljubimcima, koji piju različita pića i puše različite cigarete.

- Englez živi u crvenoj kući
- Španac ima psa
- Kafa se pije u zelenoj kući
- Ukrajinac pije čaj
- Zelena kuća je odmah desno uz belu
- Onaj koji pusi Old-Gold ima puža
- Kools se puši u žutoj kući
- Mleko se pije u srednjoj kući
- Norvežanin živi u prvoj kuci s leva
- Onaj koji puši Cesterfild živi pored onoga koji ima lisicu
- Kools se puši u kući koja je pored kuće u kojoj je konj
- Onaj koji puši Lucky pije sok od narandze
- Japanac puši Parliament
- Norvežanin živi pored plave kuće

Čija je zebra, a ko pije vodu?

k(boja, nacionalnost, cigarete, pice, kucni ljubimac)

Strukturama oblika k(boja, nacionalnost, cigarete, pice, kucni ljubimac) opisujemo date činjenice, a u listi su kuce poređane jedna pored druge, tako da po redosledu u listi imamo informaciju da li je kuća desno od kuće i da li su kuće jedna pored druge .

```
kuce(L):-L=[k(_,norvezanin,_,_,_),k(plava,_,_,_,_),k(_,_,_,mleko,_),k(_,_,_,_,_),k(_,_,_,_,_)],
        clan(k(crvena, englez,_,_,_),L),
        clan(k(_, spanac,_,_,pas),L),
%       clan(k(zela,_,_,kafa,_),L),
        clan(k(_, ukrajinac,_,caj,_),L),
        desno(k(zelena,_,_,kafa,_),k(bela,_,_,_,_),L),
        clan(k(_,_,oldgold,_,puz),L),
        clan(k(zuta,_,kools,_,_),L),
        pored(k(_,_,cesterfild,_,_),k(_,_,_,_,lisica),L),
        pored(k(_,_,kools,_,_),k(_,_,_,_,konj),L),
        clan(k(_,_,lucky,narandza,_),L),
        clan(k(_,japanac,parlaiment,_,_),L),
        clan(k(_,_,_,_,zebra),L),
        clan(k(_,_,_,voda,_),L).

problemA(X,Y):-kuce(L), clan(k(_,X,_,voda,_),L), clan(k(_,Y,_,_,zebra),L).
```

```
pored(X,Y,[X,Y|_]).
pored(X,Y,[Y,X|_]).
pored(X,Y,[_|R]):-pored(X,Y,R).

desno(X,Y,[Y,X|_]).
desno(X,Y,[_|R]):-desno(X,Y,R).
```

```
% Definiši predikat kojim se proverava da li je X clan liste.
clan(X,[X|_]).
clan(X,[_|R]):-clan(X,R).
```

IZLAZ

```
1 ?- kuce(X).
X = [k(zuta, norvezanin, kools, voda, lisica), k(plava, ukrajinac,
cesterfild, caj, konj), k(crvena, englez, oldgold, mleko, puz), k(bela,
spanac, lucky, narandza, pas), k(zelena, japanac, parlaiment, kafa, zebra)] .
```

```
% VERZIJA 2: Who owns the zebra?
%color, nationality, pets, drink, ciggarette - order

%desno_od(X, Y, L) - X desno od Y u listi L
desno_od(X, Y, [Y, X|R]).
desno_od(X, Y, [G|R]):- desno_od(X, Y, R).

pored(X, Y, L):- desno_od(X, Y, L).
pored(X, Y, L):- desno_od(Y, X, L).

sadrzi(X, [X|_]).
sadrzi(X, [G|R]):- sadrzi(X, R).
```

```
resi1(L):- L=[[_, norwegian, _, _, _], [blue, _, _, _, _], [_, _, _, milk,
_], _, _],
           sadrzi([red, english, _, _, _], L),
           sadrzi([_, spaniard, dog, _, _], L),
           sadrzi([green, _, _, coffee, _], L),
           sadrzi([_, ukrainian, _, tea, _], L),
                  desno_od([green, _, _, _, _], [ivory, _, _, _, _], L),
                  sadrzi([_, _, snail, _, oldGold], L),
                  sadrzi([yellow, _, _, _, kools], L),
                  pored([_, _, _, _, chesterfield], [_, _, fox, _, _], L),
                  pored([_, _, _, _, kools], [_, _, horse, _, _], L),
           sadrzi([_, _, _, orange, lickyStrike], L),
           sadrzi([_, japanese, _, _, parliament], L),
           sadrzi([_, _, _, water, _], L),
           sadrzi([_, _, zebra, _, _], L).

%unija dve liste
unija(L1, L2, L):- spoji(L1, L2, L3), izbaciduple(L3, L).

%presek listi
presek([X|R], L2, [X|R1]):- presek(R, L2, R1),
                             sadrzi(L2, X), !.
presek([_|R], L2, R1):- presek(R, L2, R1).
presek([], _, []).

%razlika listi
razlika([], _, []).
razlika([X|R], L2, L3):- sadrzi(L2, X), razlika(R, L2, L3), !.
razlika([X|R], L2, [X|R1]):- razlika(R, L2, R1).
```

8. %Napisati PROLOG program koji resava sledecu zagonetku. Cetiri gospodje se sastaju
%svakog cetvrtka da igraju bridz. Svaki put se dogovaraju ko ce sta da donese sledeci
%put.
%Gospodja Andric ce doneti cokoladnu tortu.
%Ni gospodja Brankovic, ni Vladislava nece doneti kolacice.
%Ruska, koja nije Davidovic, ce doneti kafu.
%Marija nece doneti vino.
%Kako se koja gospodja zove i ko sta doneti slede ce nedelje?

```
% ime, prezime, donosi
resi5(L):- L = [_, _, _, _],
           clan([_, andric, cokoladnaT], L),
           clan([ruska, _, kafa], L),
           clan([marija, _, vino], L),
           clan([ana, petrovic, _], L),
           clan([_, brankovic, _], L),
           clan([vladislava, _, _], L),
           clan([_, davidovic, _], L),
           clan([_, _, kolacice], L),
           \+(clan([ruska, davidovic, _], L)),
           \+(clan([_, brankovic, kolacice], L)),
           \+(clan([vladislava, _, kolacice],L)).
```

9. **Svakog vikenda, Mika cuva petoro komšijske dece. Deca se zovu Kata, Lazar, Marko, Nevenka Ognjen, a prezivaju Filipovic, Grbovic, Hadzic, Ivanovic i Jankovic. Svi imaju razlicit broj godina od dve do sest.**

Kako se ko zove i koliko ima godina?

- **Jedno se dete zove Lazar Jankovic.**
- **Kata je godinu dana stariji od deteta koje se preziva Ivanovic koje je godinu dana starije od Nevenke**
- **Dete Filipovic je tri godine starije od Marka.**
- **Ognjen je duplo stariji od deteta Hadzic.**

d(ime, prezime, godina)

Kako deca imaju različit broj godina u listi L smo postavili 5 struktura d(ime, prezime, godine), tako što smo godine postavili na date različite vrednosti.

```
deca(L):-L=[d(_,_,2),d(_,_,3),d(_,_,4),d(_,_,5),d(_,_,6)],
        clan(d(lazar,jankovic,_),L),
        clan(d(kata,_,G1),L),
        clan(d(_,ivanovic,G2),L),
        clan(d(nevenka,_,G3),L),
        clan(d(_,filipovic,G4),L),
        clan(d(marko,_,G5),L),
       clan(d(ognjen,_,G6),L),
        clan(d(_,hadzic,G7),L),
      clan(d(_,grbovic,_),L),
      G1=:=G2+1, G2=:=G3+1,G4=:=G5+3,G6=:=2*G7.

% Definiši predikat kojim se proverava da li je X clan liste.
clan(X,[X|_]).
clan(X,[_|R]):-clan(X,R).
```

IZLAZ

```
1 ?- deca(X).
X = [d(marko, hadzic, 2), d(nevenka, grbovic, 3), d(ognjen, ivanovic, 4), d(kata, filipovic, 5),
d(lazar, jankovic, 6)]
```

10. Četiri studenta su se takmicila u biciklizmu i plivanju:

- Andrej nije pobedio ni u jednom takmicenju.
- Onaj ko je pobedio u plivanju je bio treci u biciklizmu
- Andrej je bio bolji od Karla u plivanju, ali Karlo je bio bolji u biciklizmu
- Karlo nikada nije bio poslednji
- Darko je pobedio u biciklizmu ali Sasa je bio bolji od njega u plivanju.

Koji je bio raspored u oba takmicenja?

```
takmicenje(P,B):-P=[S1,_,_,S2],B=[darko,_,S1,S3],
      preUlisti(andrej,karlo,P),
      preUlisti(karlo,andrej,B),
      preUlisti(sasa,darko,P),
      not(S1=andrej),not(S2=karlo),not(S3=karlo).

preUlisti(X,Y,[X|R]):-clan(Y,R).
preUlisti(X,Y,[_|R]):-preUlisti(X,Y,R).
```

11. Cetiri para je doslo na maskenbal. Markova zena se maskirala kao macka. Kada su oni stigli, dva para su vec bila tamo, a jedan muskarac je bio maskiran u medveda. Prvi koji je stigao nije bio Vasa, ali je stigao pre onoga koji je bio maskiran u princa. Vestica (ne Bojana) je udata za Peru, koji se maskirao kao Paja patak. Marija je dosla posle Laze, a oboje su stigli pre Bojane. Ciganka je stigla pre Ane, pri cemu nijedna od njih nije bila udata za Betmena. Ako znamo da je zena maskirana u Snezanu stigla posle Ivane, kako je bio obucen koji par? Napisati PROLOG predikat koji odreduje resenje.

```
maskenbal(L):-
      L=[par(Muz1,_,_,_),par(_,_,_,_),par(marko,_,_,macka),par(_,_,_,_)],
       clan(par(vasa,_,_,_),L),
       clan(par(pera,paja_patak,Z,vestica),L),
       clan(par(_,betmen,Zena,Maska),L),
       preUlisti(par(_,medved,_,_),par(marko,_,_,_),L),
       preUlisti(par(Muz1,_,_,_),par(_,princ,_,_),L),
       preUlisti(par(laza,_,_,_),par(_,_,marija,_),L),
       preUlisti(par(laza,_,_,_),par(_,_,bojana,_),L),
       preUlisti(par(_,_,marija,_),par(_,_,bojana,_),L),
       preUlisti(par(_,_,_,ciganka),par(_,_,ana,_),L),
          preUlisti(par(_,_,ivana,_),par(_,_,_,snezana),L),
      not(Muz1=vasa),not (Z=bojana),not (Zena=ana),not(Maska=ciganka).
```

12. Odredi put skakača na tabli NxN počev od pozicije (X,Y) tako da na svako polje stane tacno jedanput. Ispisati redom pozicije na kojim se nalazi skakač.

```
obilazak(X,Y,N):-resenje(X, Y, N, 1, [koo(X,Y)], L), ispis(L).
resenje(_,_,N,I,L,L):-I=:=N*N.
resenje(X,Y,N, I,T,L):-I<N*N, potez(Dx,Dy), X1 is X+Dx, Y1 is Y+Dy,
                    ispravan(X1,Y1,T,N), I1 is I+1,
                    resenje(X1,Y1,N, I1, [koo(X1,Y1)|T],L).

potez(2,1). potez(2,-1). potez(1,2). potez(1,-2).
potez(-2,1). potez(-2,-1). potez(-1,2). potez(-1,-2).

ispravan(A,B,T,N):-A>0, B>0, A=<N,B=<N, not(element(koo(A,B),T)).
element(X,[X|R]).
element(X,[G|R]):-element(X,R).

ispis([G]):- write(G),!.
ispis([G|R]):-ispis(R),write(->), write(G).
```

1.4. Operatori – unifikabilno jednaki vs identicno jednaki

1. Aritmetičke operacije u Prologu:
sabiranje +, oduzimanje -,
množenje *, deljenje /,
celobrojno deljenje //, ostatak pri deljenju mod.

Operator **=** i njegova negacija \=:
Cilj Term1=Term2 uspeva ukoliko se Term1 i Term2 mogu unifikovati. Cilj Term1 \= Term2 uspeva ako cilj Term1=Term2 pada i obrnuto. Dejstvo Term1\=Term2 je isto kao not (Term1=Term2).

```
?- X=1+2
X=1+2
yes
?- X=1+2, X=3
no
```

```
?- X=1+2, Y=X*5
X=1+2
Y=(1+2)*5
yes
```

Ukoliko želimo da PROLOG interpreter izračuna izraz onda koristimo predikat **is**. Cilj
Vrednost is Aritmetički_izraz
izvodi se u dve etape:

1. izračunava se vrednost aritmetičkog izraza navedenog na desnoj strani
2. pokušava se izvršiti unifikacija izračunate vrednosti sa termom Vrednost. Cilj uspeva ako obe etape uspeju.

```
?- X is 1+2
X=3
yes

?- X is Y+2
!Eror

?- X is 1+2, X=3
X=3
yes

?- X is 5+2, X=1+6
No

?- X is 1+2, Y is X*5
X=3
Y=15

?- 3+4 is 4+3
no
```

VAZNO

```
?- N=1, N is N+1
No
?- N=1, N1 is N+1
N=1
N1=2
yes
```

Operatori poređenja U PROLOG-u:

jednakost po vrednosti =:=, negacija jednakosti =\=

veće >, veće ili jednako >=,

manje <, manje ili jednako =<.

Opšti oblik korišćenja operatora poređenja je:

Aritmetički_izraz1 operator Aritmetički_izraz2.

Izračunavaju se vrednosti aritmetičkih izraza i onda se te vrednosti upoređuju.

Za poređenje termova u PROLOG-u se koristi operator == , njegova negacija je \==.

2.

```
%razlika izmedju = (unifikabilni), \= (nisu unifikabilni), ==(identicno jednaki termovi),
\== (nisu identicno jednaki termovi)

op11(X, Y):- X = Y.

op12(X, Y):- X == Y.

%razlika izmedju is (aritmeticko izracunavanje) i =:= (aritmeticki jednaki); =\= -
aritmeticki nisu jednaki
%op1(3,4).
%op1(3,3).
%op1(X,4).
%op1(4,X).
%op1(X,3*3).
%op1(3*3, 9).

op1(X, Y):- X is Y.

op2(X, Y):- X =:= Y.
```

3. **Primer koriscenja fail predikata**

```
happy(harry).
happy(ron).
happy(hermione).
happy(hagrid).

write_all_happy :- happy(X), write(X),nl, fail.
```