```
/* Baza de cunostinte, formata din clauze:

-----> Fapte:
fapt.
fapt(Var1,Var2,...,VarN).
fapt(_var1,_Var2,...,__VarN).
fapt(arg1,arg2,...,argN).

-----> Reguli:
fapt(arg1,arg2,...,argN) :- fapt1(...), fapt2(...),..., faptM(...).
Neck: :-
Si: ,
Sau: ;
Not: \+ , not
\+ (...)
not(...)

-----> Clauze scop:
?- fapt(arg1,arg2,...,argN).
*/

/*
?- 2025**150.
?- Cat = 2025**150.
    **
   /  \
 2025 150
?- Cat is 2025**150.
?- Cat is 20.25**150.

    -
```

```
   / \
  N   1

?- bradut(5).
     *
    * *
   * * *
  * * * *
 * * * * *
*/

test(Text) :- write(Text), tab(3), write(***), nl, write(Text).

/* linie(K,N) :- puncte(K), stelute(N), nl.

puncte(0).
puncte(K) :- K>0, write('.'), PK is K-1, puncte(PK). */

/* linie(K,N) :- spatii(K), stelute(N), nl.

spatii(0).
spatii(K) :- K>0, write(' '), PK is K-1, spatii(PK). */

%%% Bradutul:

stelute(0).
stelute(N) :- N > 0, PN is N-1, stelute(PN), write(*), tab(1).

linie(K,N) :- tab(K), stelute(N), nl.

bradut(N) :- auxbradut(N,0).
```

```
auxbradut(0,_).
auxbradut(N,K) :- N>0, PN is N-1, SK is K+1, auxbradut(PN,SK), linie(K,N).

%%% Interogati: ?- bradut(15).

/* Liste:
lista vida: constanta []
listele nevide: cu operatorul binar [|]: [Head|Tail], unde Tail e o lista

[1,2,3|[4,5]] = [1,2,3,4,5] = [1|[2,3,4,5]] = [1|[2|[3|[4|[5|[]]]]]]
*/

/* In documentatia Prolog-ului:
        argumentele precedate de + trebuie furnizate in interogari;
        argumentele precedate de - vor fi calculate de Prolog in interogari;
        argumentele precedate de ? pot avea oricare dintre rolurile de mai sus in interogari.
*/

/* length(?Lista,?Lungime).
Sa scriem un predicat lungime(+Lista,-Lungime) echivalent cu predicatul predefinit length;
acesta va functiona si in modul lungime(?Lista,?Lungime). */

lungime([],0).
lungime([_|T],N) :- lungime(T,K), N is K+1.

/* Interogati:
?- lungime([1,2,3],CeLungime).
?- lungime(CeFelDeLista,3).
*/
```

```
lungimea([],0).
lungimea([_|T],N) :- lungimea(T,K), N = K+1.

/* = semnifica unificare; \= inseamna nonunificare
?- lungimea([1,2,3],Cat).
    lungimea              lungimea
    /      \              /     \
 [|]       Cat    \=     []      0
 / \
1  [|]
   / \
  2  [|]
     / \
    3  []
Rezultatul interogarii anterioare este:
Cat=  +
     / \
    +   1
   / \
  +   1
 / \
0   1
Ce inseamna unificare (orientativ):
?- f(a1,a2,...,an)=g(b1,b2,...,bk).
    f        =   g
 / | ... \     / |...\
a1 a2 ... an  b1 b2...bk
<=> f=g si n=k si a1=b1 si a2=b2 si...si an=bn
Are loc unificarea f(a1,a2,...,an)=g(b1,b2,...,bk) daca si numai daca:
   numele de operatori f si g coincid,
   operatorii f si g au aceeasi aritate (i.e. numar de argumente) n=k
```

```
   si, recursiv, argumentele ai si bi unifica pentru fiecare i intre 1 si n
*/

% Prolog-ul permite supraincarcarea operatorilor:

f :- write(*).
f(X) :- write(X).
f(10) :- write(20).
f(X,Y) :- X=Y, write(X).

/* Interogati:
?- f.
?- f(10,20).
?- f(10,10).
?- f(X,10).
?- f(10).
*/

% append(?L1,?L2,?L): predicat predefinit pentru concatenare de liste
% Sa scriem un predicat concat(?L1,?L2,?L) echivalent cu predicatul predefinit append:

concat([],L,L).
concat([H|T],L,[H|M]) :- concat(T,L,M).

/* Interogati:
?- concat([1,2],[30,400,5000],L).
?- concat([1,2],CuCeLista,[1,2,3,4,5]).
?- concat(CeLista,[3,4,5],[1,2,3,4,5]).
?- concat([1,2],CuCeLista,[A,B,3,4,5]).
?- concat(CeLista,CuCeLista,[1,2,3,4,5]).
si, la aceasta ultima interogare, cereti toate solutiile, cu:
```

```
   ; in Prolog-ul desktop
   Next sau unul dintre butoanele pentru urmatoarele 10,100,1000 solutii in Prolog-ul online
*/
```