

EXEMPLU 1. Factorial

1. Varianta 1

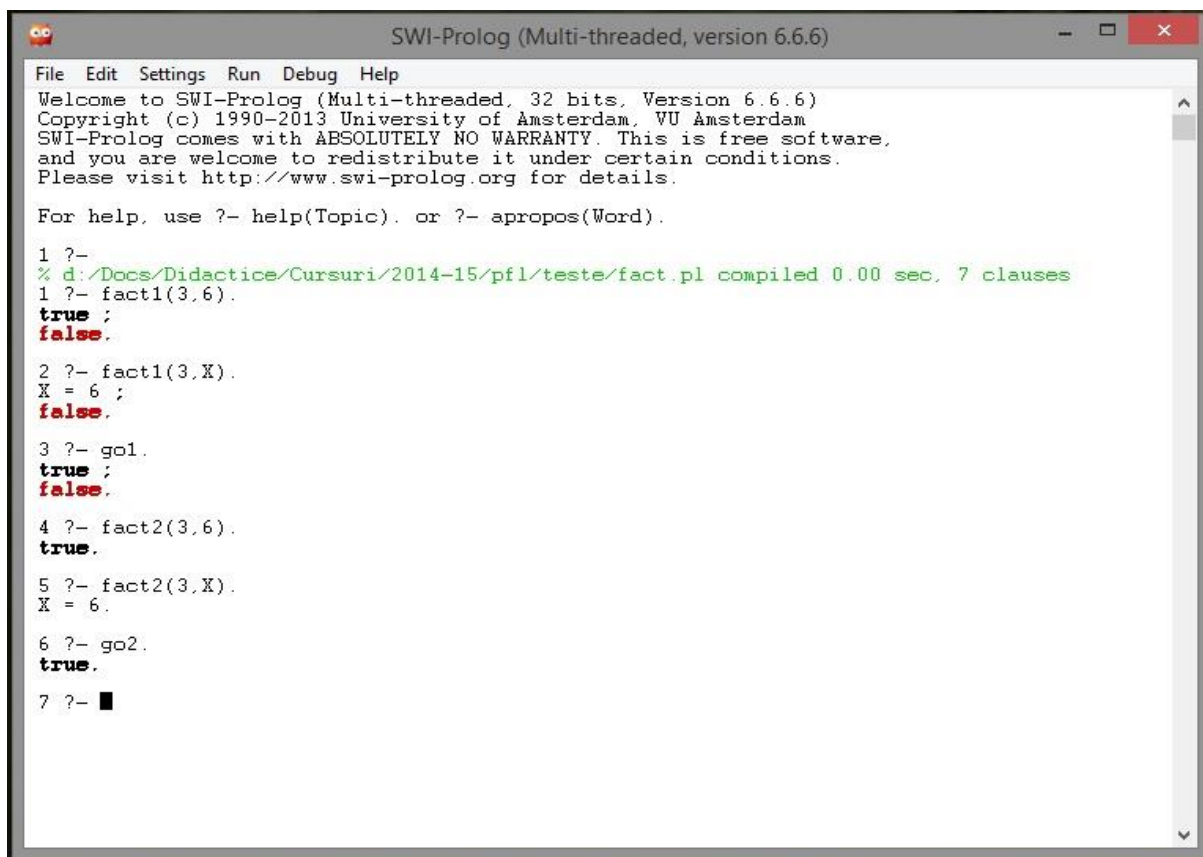
```
% (i, i) (i, o)
fact1(0, 1).
fact1(N, F) :- N > 0,
               N1 is N-1,
               fact1(N1, F1),
               F is N * F1.
```

go1 :- fact1(3, 6).

2. Varianta 2

```
% (i, i) (i, o)
fact2(0, 1) :- !.
fact2(N, F) :- N1 is N-1,
               fact2(N1, F1),
               F is N * F1.
```

go2 :- fact2(3, 6).



3. Varianta 3

```
% (i, i), (i, o)
fact3(N, F) :- N > 0,
               N1 is N-1,
               fact3(N1, F1),
               F is N * F1.
fact3(0, 1).
```

?- fact3(3, N).
N = 6.

EXEMPLU 2. Apartenență element în listă.

1. Varianta 1

```
% (i, i), (o, i)
member1(E,[E|_]).
member1(E,[_L]) :- member1(E,L).
```

```
go1 :- member1(1,[1,2,1,3,1,4]).
```

2. Varianta 2

```
% (i, i), (o, i)
member2(E,[E|_]) :- !.
member2(E,[_L]) :- member2(E,L).
```

```
go2 :- member2(1,[1,2,1,3,1,4]).
```

3. Varianta 3

```
% (i, i), (o, i)
member3(E,[_L]) :- member3(E,L).
member3(E,[E|_]).
```

```
?- member3(E,[1,2,3]).
```

```
E=3 ;
```

```
E=2 ;
```

```
E=1.
```

```
?-member3(4, [1,2,3]).
```

```
false.
```

```
?- member3(2,[1,2,3]).
```

```
true ;
```

```
false.
```

EXEMPLU 3.

Se dă o listă numerică. Se cere să se determine....

a) Fără variabilă colectoare

```
%f(L:list, Rez: list)
% (i,o) - determ
f([],[]).
f([H|T],[H|Rez]):-H mod 2==0,
    !,
    f(T,Rez).
f([_|T],Rez):-f(T,Rez).
```

b) Cu variabilă colectoare

```
%f(L:list, Rez: list)
% (i,o) - determ
f(L,Rez):-f_aux(L,Rez,[]).

%f_aux(L:list, Rez: list, Col: list)
% (i,o,i)
f_aux([],Rez,Rez).
f_aux([H|T],Rez,Col):-H mod 2==0,
    !,
    adaugsf(H,Col,Col1),
    f_aux(T,Rez,Col1).
f_aux([_|T],Rez,Col):-f_aux(T,Rez,Col).
```

EXEMPLU 4.

Se dă o listă eterogenă formată din numere, simboluri sau liste de numere. Se cere să se determine suma numerelor din lista eterogenă.

```
?- suma([1,a,[1,2,3],4],S).
S = 11.
```

```
?- suma([a,b,[],],S).
S=0.
```

```
%(L:list of numbers, S: number)
% (i,o) - determ
sumalist([],0).
sumalist([H|T],S) :- sumalist(T,S1),
    S is S1+H.
```

```
%(L:list, S: number)
% (i,o) - determ
suma([],0).
suma([H|T],S):-number(H),
    !,
```

```

        suma(T,S1),
        S is H+S1.
suma([H|T],S):-is_list(H),
        !,
        sumalist(H,S1),
        suma(T,S2),
        S is S1+S2.
suma([_|T],S):-suma(T,S).

```

EXEMPLU 5.

Se dă o listă numerică. Se cere să se determine....

```

% sP(L:list of numbers, L: list of number)
% (i,o) – nondeterm
sP([],[]).
sP([_|T],S):-sP(T,S).
sP([H|T],[H|S]):- H mod 2 =:=0,
        !,
        sP(T,S).
sP([H|T],[H|S]):-sI(T,S).

```

```

% sI(L:list of numbers, L: list of number)
% (i,o) – nondeterm
sI([H],[H]):-H mod 2 =\=0, !.
sI([_|T],S):-sI(T,S).
sI([H|T],[H|S]):-H mod 2 =:=0,
        !,
        sI(T,S).
sI([H|T],[H|S]):-sP(T,S).

```

EXEMPLU 6. (evitare apel repetat)

I. Se dă o listă numerică. Să se dea o soluție pentru evitarea apelului recursiv repetat.

```

% f(L:list of numbers, E: number)
% (i,o) – determ
f([E],E).
f([H|T],Y):- f(T,X),
        H=<X,
        !,
        Y=H.
f([_|T],X):- f(T,X).

```

II. Să se dea o soluție pentru evitarea apelului recursiv repetat.

```

% f(K:number, X:number)
% (i,o) – determ
f(1,1):-!.
f(2,2):-!.
f(K,X):- K1 is K-1,
        f(K1, Y),
        Y>1,
        !,
        K2 is K-2,
        X=K2.
f(K,X):- K1 is K-1,
        f(K1, X).

```

EXEMPLU 7.

Se dă o listă numerică. Se cere să se afișeze elementele listei în ordine crescătoare. Se va folosi sortarea arborescentă (folosind un ABC).

Indicație. Se va construi un ABC cu elementele listei. Apoi, se va parcurge ABC în inordine.

```
% domeniul corespunzător ABC – domeniu cu alternative
% arbore=arb(integer, arbore, arbore);nil
% Functorul nil îl asociem arborelui vid

% (integer, arbore, arbore) – (i,i,o) determ
% insereaza un element într-un ABC
inserare(E,nil,arb(E,nil,nil)).
inserare(E,arb(R,S,D),arb(R,SNou,D)):-E<R,! ,inserare(E,S,SNou).
inserare(E,arb(R,S,D),arb(R,S,DNou)):-inserare(E,D,DNou).

% (arbore) – (i) determ
% afișează nodurile arborelui în inordine
inordine(nil).
inordine(arb(R,S,D)):-inordine(S),write(R),nl,inordine(D).

% (arbore, list) – (i,o) determ
% creează un ABC cu elementele unei liste
creeazaArb([],nil).
creeazaArb([H|T],Arb):-creeazaArb(T,Arb1),inserare(H,Arb1,Arb).

% (list) – (i) determ
% afișează elementele listei în ordine crescătoare (folosind sortare arborescentă)
sortare(L):-creeazaArb(L,Arb),inordine(Arb).
```

EXEMPLU 8.

PROBLEMA CELOR 3 CASE.

1. Englezul locuiește în prima casă din stânga.
2. În casa imediat din dreapta celei în care s află lupul se fumează Lucky Strike.
3. Spaniolul fumează Kent.
4. Rusul are cal.

Cine fumează LM? Al cui este câinele?

```

SWI-Prolog -- d:/Gabi/gabi/FACULTAT/2014-2015/PLF/DOC/Exemple_SWI/exemple.pl
File Edit Settings Run Debug Help
% library(win_menu) compiled into win_menu 0.00 sec, 29 clauses
% c:/users/istvan/appdata/roaming/swi-prolog/pl.ini compiled 0.00 sec, 1 clauses
% d:/Gabi/gabi/FACULTAT/2014-2015/PLF/DOC/Exemple_SWI/exemple.pl compiled 0.00 sec, 95 clauses
Welcome to SWI-Prolog (Multi-threaded, 32 bits, Version 6.2.0)
Copyright (c) 1990-2012 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

1 ?- rezolva(N,A,T).
N = [eng, spa, rus],
A = [caine, lup, cal],
T = [lm, kent, ls] ;
N = [eng, rus, spa],
A = [lup, cal, caine],
T = [lm, ls, kent] ;
false.

2 ?- █

```

% rezolva - (o,o,o)

% candidati - (o,o,o)

% restrictii - (i,i,i)

rezolva(N,A,T):-candidati(N,A,T), restrictii(N,A,T).

candidati(N,A,T):-perm([eng,spa,rus],N),
perm([caine,lup,cal],A),
perm([lm,kent,ls],T).

restrictii(N,A,T):-aux(N,A,T,eng,_,_,1),
aux(N,A,T,_,lup,_,Nr),
dreapta(Nr,M),
aux(N,A,T,_,_,ls,M),
aux(N,A,T,spa,_,kent,_),
aux(N,A,T,rus,cal,_,_).

% dreapta - (i,o)

dreapta(I,J):-J is I+1.

% aux (i,i,i,o,o,o,o)

aux([N1,_,_],[A1,_,_],[T1,_,_],N1,A1,T1,1).
aux(,N2,_,,A2,_,,T2,_,N2,A2,T2,2).
aux(,_,N3,_,,A3,_,,T3,_,N3,A3,T3,3).

% insereaza (i,io)

insereaza(E,L,[E|L]).

insereaza(E,[H|L],[H|T]):-insereaza(E,L,T).

% perm (i,o)

perm(,[]).

perm([H|T],L):-perm(T,P),insereaza(H,P,L).

PROBLEMA CELOR 5 CASE.

5 people live in the five houses in a street. Each has a different profession, animal, favorite drink, and each house is a different color.

1. The Englishman lives in the red house
2. The Spaniard owns a dog
3. The Norwegian lives in the first house on the left
4. The Japanese is a painter
5. The green house is on the right of the white one
6. The Italian drinks tea
7. The fox is in a house next to the doctor
8. Milk is drunk in the middle house
9. The horse is in a house next to the diplomat
10. The violinist drinks fruit juice
11. The Norwegians house is next to the blue one
12. The sculptor breeds snails
13. The owner of the green house drinks coffee
14. The diplomat lives in the yellow house

Who owns the zebra? Who drinks water?

EXEMPLU 9

Se dă o mulțime de numere naturale nenule reprezentată sub forma unei liste. Să se determine toate posibilitățile de a scrie un număr N dat sub forma unei sume a elementelor din listă.

```
% list=integer*
% candidat(list, integer) (i, o) - nedeterminist
% un element posibil a fi adaugat in lista solutie

candidat([E|_],E).
candidat([_|T],E):-candidat(T,E).

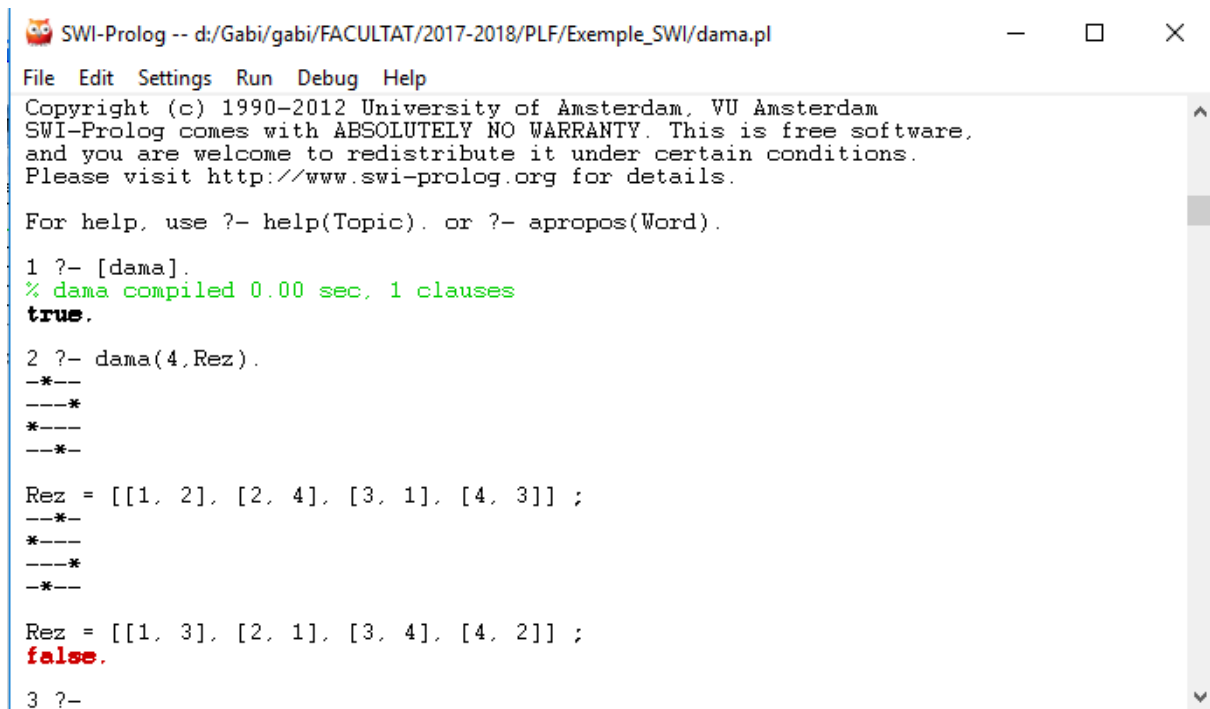
% solutie(list,integer,list) (i,i,o) nedeterminist
% solutie_aux(list,integer,list,list,integer) (i,i,o,i,i) nedeterminist
% al patrulea argument colecteaza solutia, al cincilea argument
% reprezinta suma elementelor din colectoare

solutie(L,N,Rez):-candidat(L,E), E=<N, solutie_aux(L,N,Rez,[E],E).

solutie_aux(_,N,Rez,Rez,N):-!.
solutie_aux(L,N,Rez,[H|Col],S):-candidat(L,E),
    E<H,
    S1 is S+E,
    S1=<N,
    solutie_aux(L,N,Rez,[E|[H|Col]],S1).
```

EXEMPLU 10

Să se dispună N dame pe o tablă de șah NxN, încât să nu se atace reciproc.



```
SWI-Prolog -- d:/Gabi/gabi/FACULTAT/2017-2018/PLF/Exemple_SWI/dama.pl
File Edit Settings Run Debug Help
Copyright (c) 1990-2012 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

1 ?- [dama].
% dama compiled 0.00 sec, 1 clauses
true.

2 ?- dama(4, Rez).
---*
---*
*---
---*

Rez = [[1, 2], [2, 4], [3, 1], [4, 3]] ;
---*
*---
---*
---*

Rez = [[1, 3], [2, 1], [3, 4], [4, 2]] ;
false.

3 ?-
```

% (integer,list*) – (i, o) nedeterm.

```
dama(N,Rez):-candidat(E,N),
             dama_aux(N,Rez,[[N,E]],N),
             tipar(N,Rez).
```

% (integer,integer) – (o, i) nedeterm.

```
candidat(N,N).
candidat(E,I):-I>1, I1 is I-1, candidat(E,I1).
```

% (integer,list*,list*,integer) – (i,o, i, i) nedeterm.

```
dama_aux(_,Rez,Rez,1):-!.
dama_aux(N,Rez,C,Lin):-
    candidat(Col1,N),
    Lin1 is Lin-1,
    valid(Lin1,Col1,C),
    dama_aux(N,Rez,[[Lin1,Col1]|C],Lin1).
```

% (integer,integer,list*) – (i,i, i) determ.

```
valid(_,_,[]).
valid(Lin,Col,[[Lin1,Col1]|T]):-
    Col=\=Col1,
    DLin is Col-Col1,
    DCol is Lin-Lin1,
    abs(DLin)=\=abs(DCol),
    valid(Lin,Col,T).
```



```
% (integer, list*) – (i,i) determ.  
tipar(_,[ ]):-nl.  
tipar(N,[[_ ,Col]|T]):-tipLinie(N,Col),tipar(N,T).
```

```
% (integer, char) – (i,o) determ.  
caracter(1,'*'):-!.  
caracter(_,'-').
```

```
% (integer, list*) – (i,i) determ.  
tipLinie(0,_):-nl,!.  
tipLinie(N,Col):-caracter(Col,C),  
                write(C),  
                N1 is N-1,  
                Col1 is Col-1,  
                tipLinie(N1,Col1).
```