

Laborator 2 – Prolog 1

Implementations will be done in:

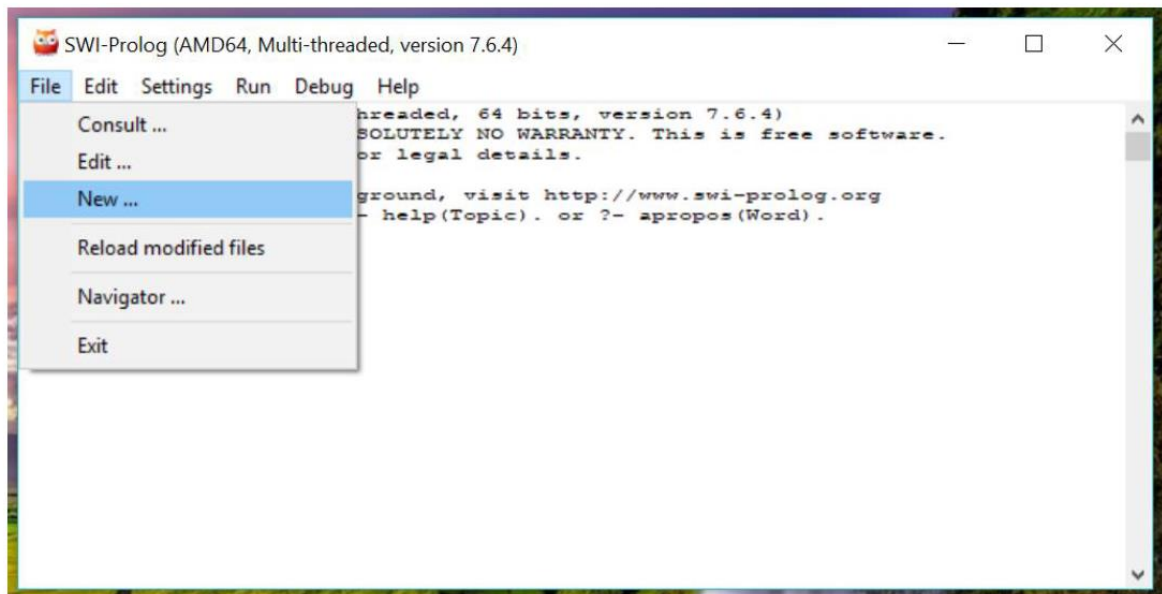
- SWI Prolog (de instalat)

or

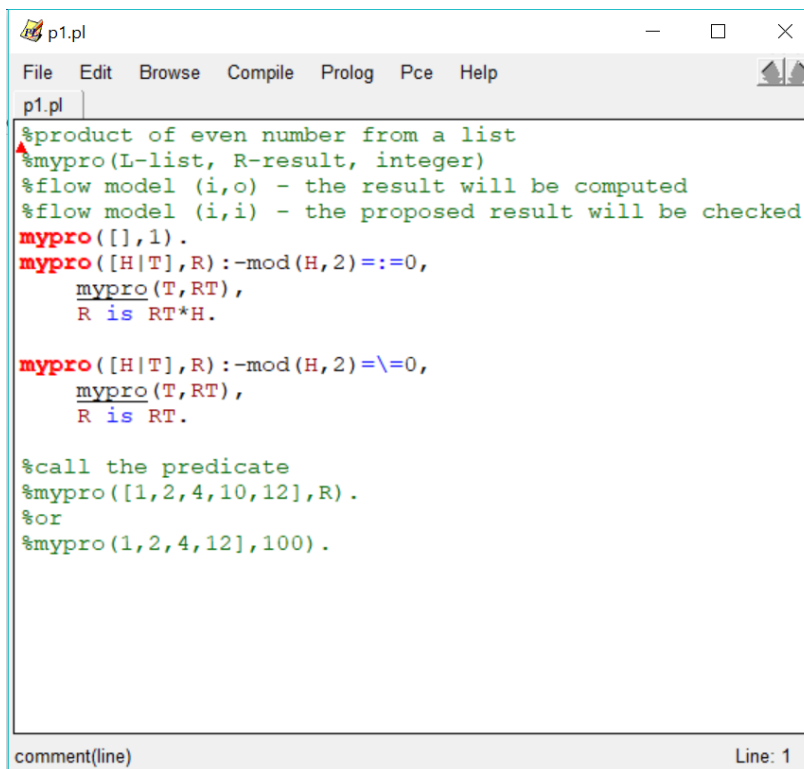
- SWISH Prolog (online)

Problema 1 – model in **SWI PROLOG**

Download from here: <http://www.swi-prolog.org/Download.html> and install.



Edit your code:



The screenshot shows a window titled 'p1.pl' with a menu bar (File, Edit, Browse, Compile, Prolog, Pce, Help). The code in the editor is as follows:

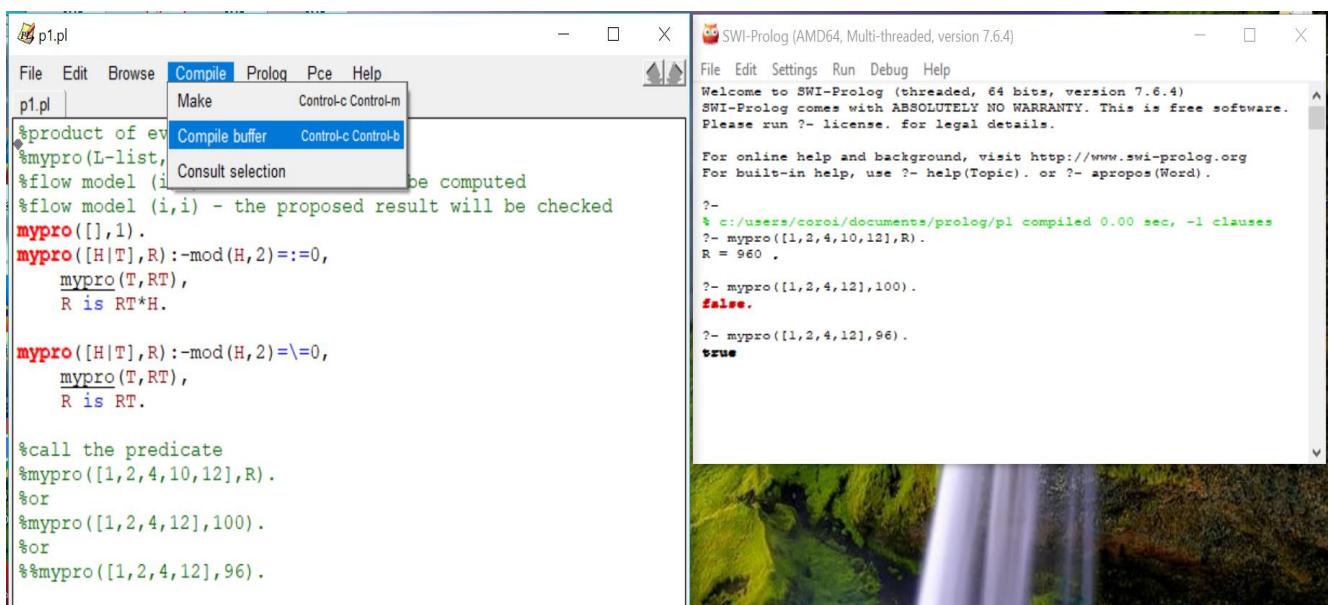
```
%product of even number from a list
%mypro(L-list, R-result, integer)
%flow model (i,o) - the result will be computed
%flow model (i,i) - the proposed result will be checked
mypro([],1).
mypro([H|T],R):-mod(H,2)=:=0,
    mypro(T,RT),
    R is RT*H.

mypro([H|T],R):-mod(H,2)=\=0,
    mypro(T,RT),
    R is RT.

%call the predicate
%mypro([1,2,4,10,12],R).
%or
%mypro(1,2,4,12,100).
```

At the bottom, there is a status bar with 'comment(line)' on the left and 'Line: 1' on the right.

Compile (1) and run (2)



The screenshot shows two windows. The left window is the 'p1.pl' editor, with the 'Compile' menu open, showing options: 'Make', 'Compile buffer', 'Consult selection', 'Control-c Control-m', and 'Control-c Control-b'. The code in the editor is the same as in the previous screenshot.

The right window is the 'SWI-Prolog (AMD64, Multi-threaded, version 7.6.4)' console. It displays the following text:

```
Welcome to SWI-Prolog (threaded, 64 bits, version 7.6.4)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit http://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?-
% c:/users/coroi/documents/prolog/pl compiled 0.00 sec, -1 clauses
?- mypro([1,2,4,10,12],R).
R = 960 .

?- mypro([1,2,4,12],100).
false.

?- mypro([1,2,4,12],96).
true
```

At the bottom of the console window, there is a small image of a waterfall.

Trace for activate step by step execution:



SWI-Prolog (AMD64, Multi-threaded, version 7.6.4)

— □ ×

File Edit Settings Run Debug Help

```
?- trace.  
true.  
  
[trace] ?- mypro([1,2,4,12],100).  
Call: (8) mypro([1, 2, 4, 12], 100) ? creep  
Call: (9) 1 mod 2=:0 ? creep  
Fail: (9) 1 mod 2=:0 ? creep  
Redo: (8) mypro([1, 2, 4, 12], 100) ? creep  
Call: (9) 1 mod 2=\=0 ? creep  
Exit: (9) 1 mod 2=\=0 ? creep  
Call: (9) mypro([2, 4, 12], _1218) ? creep  
Call: (10) 2 mod 2=:0 ? creep  
Exit: (10) 2 mod 2=:0 ? creep  
Call: (10) mypro([4, 12], _1224) ? creep  
Call: (11) 4 mod 2=:0 ? creep  
Exit: (11) 4 mod 2=:0 ? creep  
Call: (11) mypro([12], _1230) ? creep  
Call: (12) 12 mod 2=:0 ? creep  
Exit: (12) 12 mod 2=:0 ? creep  
Call: (12) mypro([], _1236) ? creep  
Exit: (12) mypro([], 1) ? creep  
Call: (12) _1240 is 1*12 ? creep  
Exit: (12) 12 is 1*12 ? creep  
Exit: (11) mypro([12], 12) ? creep  
Call: (11) _1246 is 12*4 ? creep  
Exit: (11) 48 is 12*4 ? creep  
Exit: (10) mypro([4, 12], 48) ? creep  
Call: (10) _1252 is 48*2 ? creep  
Exit: (10) 96 is 48*2 ? creep  
Exit: (9) mypro([2, 4, 12], 96) ? creep  
Call: (9) 100 is 96 ? creep  
Fail: (9) 100 is 96 ? creep  
Redo: (11) mypro([12], _1230) ? creep  
Call: (12) 12 mod 2=\=0 ? creep  
Fail: (12) 12 mod 2=\=0 ? creep  
Fail: (11) mypro([12], _1230) ? creep  
Redo: (10) mypro([4, 12], _1224) ? creep  
Call: (11) 4 mod 2=\=0 ? creep  
Fail: (11) 4 mod 2=\=0 ? creep  
Fail: (10) mypro([4, 12], _1224) ? creep  
Redo: (9) mypro([2, 4, 12], _1218) ? creep  
Call: (10) 2 mod 2=\=0 ? creep  
Fail: (10) 2 mod 2=\=0 ? creep  
Fail: (9) mypro([2, 4, 12], _1218) ? creep  
Fail: (8) mypro([1, 2, 4, 12], 100) ? creep  
false.
```

Problema 2 – model SWISH Prolog

The screenshot shows the SWISH Prolog web interface. The browser address bar displays `https://swish.swi-prolog.org`. The interface includes a menu bar with **File**, **Edit**, **Examples**, and **Help**. A search bar is located in the top right. The main editor area contains the following Prolog code:

```
1 %sum of elements from a list L
2 %sum (L-list, S-result, integer)
3 %sum (i, o)
4
5 suma([],0).
6 suma([H|T],S):-suma(T,ST),
7   S is H+ST.
8
9
10
11
12
13
14
15
16 %call ----->
17 %ctrl+enter pt rulare
```

On the right side, a large owl logo is visible. Below it, a window displays the execution result for the query `suma([1,2,3,4,5,1,2,3],R).`:

```
R = 21
```

Below the result, a prompt `?- suma([1,2,3,4,5,1,2,3],R).` is shown.

Step by step execution – using **TRACE**, here in Swish Prolog

The screenshot shows the SWISH Prolog web interface with the same Prolog code as before. The code is:

```
1 %sum of elements from a list L
2 %sum (L-list, S-result, integer)
3 %sum (i, o)
4
5 suma([],0).
6 suma([H|T],S):-suma(T,ST),
7   S is H+ST.
8
9
10
11
12
13 %call ----->
14 %ctrl+enter pt rulare
15
16 %pentru executia pas cu pas,
17   |se foloseste TRACE
```

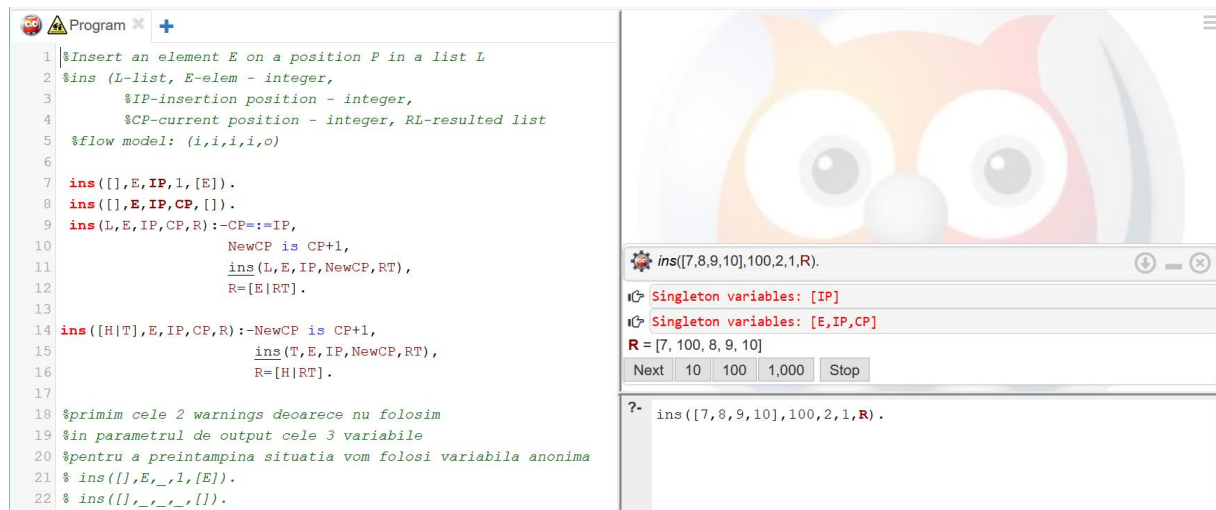
A window titled `trace, (suma([1,2,3,4,5,1,2,3],R)).` is open on the right, displaying the step-by-step execution of the query. The trace shows the following sequence of calls and exits:

```
Call: suma([1, 2, 3, 4, 5, 1, 2, 3], _3996)
Call: suma([2, 3, 4, 5, 1, 2, 3], _4272)
Call: suma([3, 4, 5, 1, 2, 3], _4272)
Call: suma([4, 5, 1, 2, 3], _4272)
Call: suma([5, 1, 2, 3], _4272)
Call: suma([1, 2, 3], _4272)
Call: suma([2, 3], _4272)
Call: suma([3], _4272)
Call: suma([], _4272)
Exit: suma([], 0)
Call: _4276 is 3+0
Exit: 3 is 3+0
Exit: suma([3], 3)
Call: _4282 is 2+3
```

Below the trace, a prompt `?- trace, (suma([1,2,3,4,5,1,2,3],R)).` is shown.

Problema 3 – model in SWISH or in SWI PROLOG, is up to you :)

(aici folosim nume de variabile)

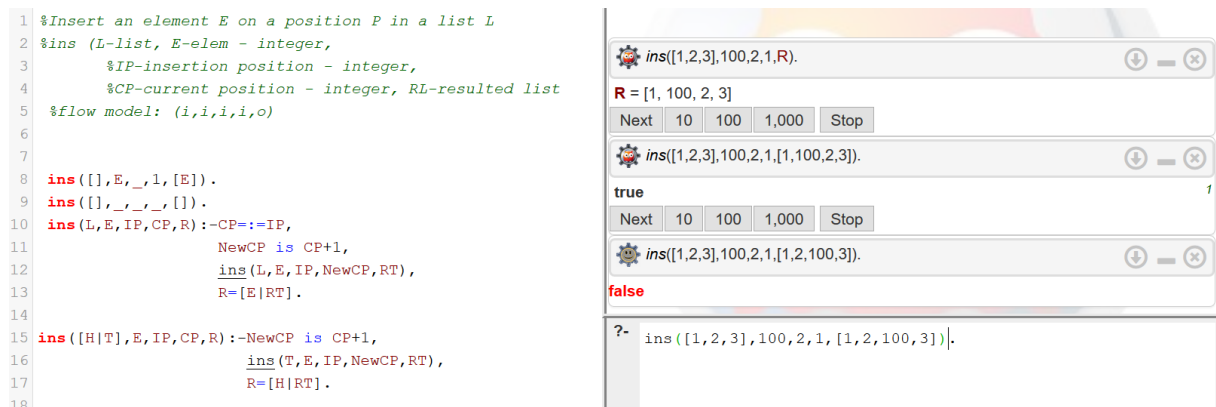


The screenshot shows a SWISH Prolog environment. On the left, a code editor displays a Prolog program for inserting an element into a list. The program includes comments in Romanian and uses a flow model. On the right, the execution window shows the query `ins([7,8,9,10],100,2,1,R).` being executed. The execution results show the singleton variables `[IP]` and `[E,IP,CP]`, and the resulting list `R = [7, 100, 8, 9, 10]`. The execution window also has buttons for 'Next', '10', '100', '1,000', and 'Stop'.

```
1 %Insert an element E on a position P in a list L
2 %ins (L-list, E-elem - integer,
3     %IP-insertion position - integer,
4     %CP-current position - integer, RL-resulted list
5 %flow model: (i,i,i,i,o)
6
7 ins([],E,IP,1,[E]).
8 ins([_,E,IP,CP,[]).
9 ins(L,E,IP,CP,R):-CP:=IP,
10     NewCP is CP+1,
11     ins(L,E,IP,NewCP,RT),
12     R=[E|RT].
13
14 ins([H|T],E,IP,CP,R):-NewCP is CP+1,
15     ins(T,E,IP,NewCP,RT),
16     R=[H|RT].
17
18 %primim cele 2 warnings deoarece nu folosim
19 %in parametrul de output cele 3 variabile
20 %pentru a preintampina situatia vom folosi variabila anonima
21 % ins([],E,_,1,[E]).
22 % ins([],_,_,[],[]).
```

Aici folosim variabila anonima:

Si verificam functionalitatea predicatului si folosind modelul de flow (i, i, i, i, i) – iar ca rezultat vom avea True/False



The screenshot shows a SWISH Prolog environment. On the left, a code editor displays a Prolog program for inserting an element into a list, using anonymous variables for the insertion position and the resulting list. On the right, the execution window shows the query `ins([1,2,3],100,2,1,R).` being executed. The execution results show the singleton variables `R = [1, 100, 2, 3]` and the resulting list `R = [1, 100, 2, 3]`. The execution window also has buttons for 'Next', '10', '100', '1,000', and 'Stop'.

```
1 %Insert an element E on a position P in a list L
2 %ins (L-list, E-elem - integer,
3     %IP-insertion position - integer,
4     %CP-current position - integer, RL-resulted list
5 %flow model: (i,i,i,i,o)
6
7
8 ins([],E,_,1,[E]).
9 ins([],_,_,[],[]).
10 ins(L,E,IP,CP,R):-CP:=IP,
11     NewCP is CP+1,
12     ins(L,E,IP,NewCP,RT),
13     R=[E|RT].
14
15 ins([H|T],E,IP,CP,R):-NewCP is CP+1,
16     ins(T,E,IP,NewCP,RT),
17     R=[H|RT].
18
```

Va rog sa va instalati si sa verificati functionalitatea predicatelor de mai sus, urmarind cel putin pentru o problema executia pas cu pas.