```prolog
% Define number and identifier literals
% num/1 - Defines numbers
num(N) :- number(N).
% id/1 - Defines identifiers
id(I) :- atomic(I).

boolean(_, true, true).
boolean(_, false, false).
boolean(S0, LE > RE, X) :-
    expression(S0, LE, LR),
    expression(S0, RE, RR),
    (LR > RR) -> (X = true) ;
                 (X = false).
boolean(S0, LE >= RE, X) :-
    expression(S0, LE, LR),
    expression(S0, RE, RR),
    (LR >= RR) -> (X = true) ;
                  (X = false).
boolean(S0, LE < RE, X) :-
    expression(S0, LE, LR),
    expression(S0, RE, RR),
    (LR < RR) -> (X = true) ;
                 (X = false).
boolean(S0, LE =< RE, X) :-
    expression(S0, LE, LR),
    expression(S0, RE, RR),
    (LR =< RR) -> (X = true) ;
                  (X = false).
boolean(S0, LE == RE, X) :-
    expression(S0, LE, LR),
    expression(S0, RE, RR),
    (LR == RR) -> (X = true) ;
                  (X = false).

%Execute given a program P and Binding Environment S0
% Execute/3 - Defines how a program should be executed.
% S0 is the binding Environment before the program is executed.
% P is the program.
% Sn is the binding environment after execution of the program.
execute(S0,P,Sn):-
    command(S0,P,Sn).

% set/2 - Defines a set relation between an identifier and a number.
set(id(I),num(E)) :- id(I), num(E).

% bind/4 - Defines the procedure of binding a set relation between an identifier and
 a number and a binding environment.
bind([], I, E, [set(I,E)]).
% Special case of bind/4 that make sure an identifier with an already existing set r
ealtion is set to the new value and not duplicated and appended.
bind([set(I,_)|S0], I,E, [set(I,E)|S0]).
bind([set(H,A)|S0], I, E, [set(H,A)|Sn]) :-
    H \= I,
    bind(S0,I,E,Sn).

% expression/3 - Defines the evaluation of arithmetic expressions.
expression(S0,id(E),R) :-
    member(set(E,R), S0). % Retrives the numeric value of an identifier already in t
he bidning environment.
expression(_,num(E),E).
% Defines the 'addition' operator for two expressions.
expression(S0, E1 + E2, R) :-
    expression(S0, E1, R1), % Evaluate expression E1.
    expression(S0, E2, R2), % Evaluate expression E2.
    R is (R1 + R2). % Assign the return value the value of E1 + E2.
% Defines the 'subtraction' operator for two expressions.
expression(S0, E1 - E2, R) :-
    expression(S0, E1, R1), % Evaluate expression E1.
    expression(S0, E2, R2), % Evaluate expression E2.
    R is (R1 - R2). % Assign the return value the value of E1 - E2.
% Defines the 'mutliplication' operator for two expressions.
expression(S0, E1 * E2, R) :-
    expression(S0, E1, R1), % Evaluate expression E1.
```

```
    expression(S0, E2, R2), % Evaluate expression E2.
    R is (R1 * R2). % Assign the return value the value of E1 * E2.
% Defines the 'negation' operator for two expressions.
expression(S0, - E, R) :-
    expression(S0, E, R1), % Evaluate expression E.
    R is ( R1 * -1). % Assign the return value the value of -E.


%Define commands
% Command/3 - Defines the evaluation of different comand structures in a program.
% Define command skip as a fact
command(S0,skip,S0).
% Defines the command set.
command(S0,set(id(I),E),Sn) :-
    expression(S0,E,R), % Evaluate expression E
    bind(S0,I,R,Sn).    % Bind the evaluate expression E to the identifier I and add
s the set to the binding environment.
% Defines the command if.
command(S0,if(B,C1,_),Sn) :-
    (boolean(S0,B,true),command(S0,C1,Sn)). % If the boolean expression B is true pe
rform command C1.
command(S0,if(B,_,C2),Sn) :-
    ( boolean(S0,B,false),command(S0,C2,Sn)). % If the boolean expression B is false
 perform command C2.
% Defines the command seq
command(S0,seq(C1,C2),Sn) :-
    command(S0,C1,Sr),  % First perform action C1
    command(Sr,C2,Sn).  % Then perform actino C2
% Defines the command while
command(S0,while(B,_),S0) :-
    boolean(S0,B,false).    % If the boolean expression B is false, stop.
command(S0,while(B,C),Sn) :-
    boolean(S0,B, true),
    command(S0,C,Sr), % Perform action C.
    command(Sr,while(B,C),Sn).  % Recursivley call the command while with the same b
oolean expression but with updated Binding environment.


% ------------------------
% ------EXAMPLE QUERY------
% ------------------------
%
% Execute the program  seq(set(id(a),num(1)), set(id(a),id(a)+num(2)))
% -----------------------------------------------
%
% ?- execute([],seq(set(id(a),num(1)), set(id(a),id(a)+num(2))),X).
% X = [set(a, 3)] ;
% no
%
% Execute program seq(set(id(y),num(1)),while(id(x) > num(1),seq(set(id(y), id(y) *
id(x)), set(id(x), id(x) - num(1)))))
% --------------------------------
% ?- execute([set(x,3)],seq(set(id(y),num(1)),while(id(x) > num(1),seq(set(id(y), id
(y) * id(x)),set(id(x), id(x) - num(1)))))),X).
% X = [set(x, 1), set(y, 6)] ;
% no
```