A01: 9,5P - wegen verspäteter Abgabe nicht wertbar

akzeptiert

A1: 3,5P

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% Aufgabenblatt 01 - SE3-LP WiSe 16/17
%
% Finn-Lasse Jörgensen
% Fabian Behrend
% Daniel Klotzsche 6535732
%
% Wir sind bereit folgende Aufgaben zu präsentieren:
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

[familie]
%consult('familie.pl'
listing(mutter_von).
```

Ergebnis? -0,5P

```
assert(mutter_von(helga,assert)).
asserta(mutter_von(helga,asserta)).
assertz(mutter_von(helga,assertz)).
%open('myClauses.pl',write,S), set_output(S), listing, close(S).

%%% A1 %%%
% 1.
% 2.
% 3.
% assert  : fügt den eintrag an der aktuell letzten stelle hinzu
% asserta : fügt den eintrag vorne an die datenbank an
% assertz : fügt den eintrag am ende in die datenbank an
```

A2: 6P

```
%%% A2 %%%
%% 1.

% a)
mutter_von(julia,otto).
% -> true

% b)
vater_von(otto,helga).
% -> true

% c)
vater_von(Vater,julia).
% -> false

% d)
vater_von(otto,Kind).
% -> Kind=hans;
%    Kind=helga.

% e)
mutter_von(M,K1);vater_von(V,K1).

% f)
not(vater_von(hans,Kind)).
% -> true

% g)
not(vater_von(johannes,Kind)).
% -> false

% h)
vater_von(otto,_).
% -> true

%% 2.
```



```
mutter_von(charlotte,Kind);(mutter_von(Kind,Enkel);vater_von(Kind,Enkel)).

%% 3.

/*
Im Trace werden die Einzelschritte angezeigt, welche von Prolog durchgeführt we
rden.
-Call: Es wird eine moegliche Belegung fuer Variablen in einer Relation gesucht
.
-Exit: Eine Moegliche Belegung wird zurückgegeben.
-Redo: Die naechste moegliche Belegung wird angefordert.
-Fail: Es wurde keine Belegung der Variable in einer Relation gefunden.
*/

trace().
mutter_von(julia,otto).
% ->    Call: (7) mutter_von(julia, otto) ? creep
% ->    Exit: (7) mutter_von(julia, otto) ? creep
% -> true.

vater_von(otto,helga).
% ->    Call: (7) vater_von(otto, helga) ? creep
% ->    Exit: (7) vater_von(otto, helga) ? creep
% -> true.

vater_von(Vater,julia).
% ->    Call: (7) vater_von(_G5833, julia) ? creep
% ->    Fail: (7) vater_von(_G5833, julia) ? creep
% -> false.

vater_von(otto,Kind).
% ->     Call: (7) vater_von(otto, _G6020) ? creep
% ->     Exit: (7) vater_von(otto, hans) ? creep
% -> Kind = hans .

mutter_von(M,K1);vater_von(V,K1).
% ->       Call: (8) mutter_von(_G6091, _G6092) ? creep
% ->       Exit: (8) mutter_von(marie, hans) ? creep
% ->    M = marie,
% ->    K1 = hans .

not(vater_von(hans,Kind)).
% ->       Call: (8) vater_von(hans, _G6050) ? creep
% ->       Fail: (8) vater_von(hans, _G6050) ? creep
% -> true.

not(vater_von(johannes,Kind)).
% ->      Call: (8) vater_von(johannes, _G6074) ? creep
% ->      Exit: (8) vater_von(johannes, klaus) ? creep
% -> false.

vater_von(otto,_).
% ->      Call: (7) vater_von(otto, _G6002) ? creep
% ->      Exit: (7) vater_von(otto, hans) ? creep
% -> true .
nodebug().
```