```
/* LABORATORIUM 2 */


/* -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  -- */
/* Zadanie 1 */

% From the book
% PROLOG PROGRAMMING IN DEPTH
% by Michael A. Covington, Donald Nute, and Andre Vellino
% (Prentice Hall, 1997).
% Copyright 1997 Prentice-Hall, Inc.
% For educational use only

% File INTERAC.PL
% Simple interactive program

capital_of(georgia,atlanta).
capital_of(florida,tallahassee).

go :- write('What state do you want to know about?'),nl,
      write('Type its name, all lower case, followed by a period.'),nl,
      read(State),
      capital_of(State,City),
      write('Its capital is: '),write(City),nl.

/* Zapytania
 * write('Ala ma '),write('kota'),nl,write('w ciapki!'). -- wypisuje tekst z nowa linia
 * go. -- uruchamia predykat go, ktory wypisuje tekst i pobiera dane od uzytkownika
 * -- wprowadzane dane nalezy zakonczyc kropka
 */
/* -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  -- */

/* -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  -- */
/* Zadanie 2 */

/*
 * family.pl
 * taken from Bratko, 3rd ed, ch.1, p.17
 *
 */

:- dynamic(kobieta/1).

rodzic(kasia,robert).
rodzic(tomek,robert).
rodzic(tomek,eliza).
rodzic(robert,anna).
rodzic(robert,magda).
rodzic(magda,jan).

kobieta(kasia).
kobieta(eliza).
kobieta(magda).
kobieta(anna).

mezczyzna(tomek).
mezczyzna(robert).
mezczyzna(jan).

potomek(X,Y) :-
    rodzic(Y,X).

matka(X,Y) :-
    rodzic(X,Y),
    kobieta(X).

ojciec(X,Y) :-
    rodzic(X,Y),
    mezczyzna(X).

dziadkowie(X,Y) :-
    rodzic(X,Z),
    rodzic(Z,Y).

dziadek(X,Y) :-
    dziadkowie(X,Y),
    mezczyzna(X).

babcia(X,Y) :-
    dziadkowie(X,Y),
    kobieta(X).

siostra(X,Y) :-
    rodzic(Z,X),
    rodzic(Z,Y),
    kobieta(X),
    X \= Y.

brat(X,Y) :-
    rodzic(Z,X),
    rodzic(Z,Y),
```

```
    mezczyzna(X),
    X \= Y.

przodek(X,Y) :-
    rodzic(X,Y).

przodek(X,Z) :-
    rodzic(X,Y),
    przodek(Y,Z).

%%
madziecko(X) :-
    rodzic(X,_).

               % 2 razy robert? unique?

%% ex 1.3
szczesciarz(X) :-
    madziecko(X).

dwojedzieci(X) :-
    rodzic(X,Y),
    siostra(_,Y).

%% ex 1.4
wnuk(X,Z) :-
    rodzic(Y,X),
    rodzic(Z,Y).

%% ex 1.5
ciocia(X,Y) :-
    rodzic(Z,Y),
    siostra(X,Z).


% From the book
% PROLOG PROGRAMMING IN DEPTH
% by Michael A. Covington, Donald Nute, and Andre Vellino
% (Prentice Hall, 1997).
% Copyright 1997 Prentice-Hall, Inc.
% For educational use only

% File CAPITALS.PL or KB.PL
% Knowledge base for several examples in Chapter 2

:- dynamic(capital_of/2).      % Remove if not needed. See text, section 2.8.

capital_of(georgia,atlanta).
capital_of(california,sacramento).
capital_of(florida,tallahassee).
capital_of(maine,augusta).




/* Zapytania:
 *
 * kobieta(K),write(K),write(' to kobieta.'),nl. -- znajduje jedna z kobiet i wypisuje jej imie,
 * -- dziala zgodnie z algorytmem prologa przeszukiwania drzewa, znajduje pierwsze rozwiazanie poprawne
 * -- i wraca
 *
 * kobieta(K),write(K),write(' to kobieta.'),nl,fail. -- wymusza odnajdywanie kolejnych kobiet w
 * -- w jednym wywolaniu, fail powoduje, ze predykat jest zawsze niespelniony i prolog wraca do poszukiwania
 * -- rozwiazania od nowa zmieniajac wartosc K
 *
 * kobieta(K),fail. -- zwraca false ponieważ ostateczna wartosc predykatu to false, wynika to z tego, że
 * -- ostatecznie prolog nie znalzl ani jednego rozwiazania ktore zwrociloby true z powodu dodania .fail
 * -- na koncu
 *
 * capital_of(A,B), write(B), write(' to stolica '), write(A), nl. -- prolog szuka pierwszego rozwiazaia
 * -- w drzewie, znajduje pierwsze ktore jest zapisane czyli 'georgia', daje ono true, szuka wiec czegos
 * -- aby podstawic za B i co laczy sie z 'georgia' czyli atlanta.
 *
 * capital_of(A,B), write(B), write(' to stolica '), write(A), nl, fail. -- prolog wypisuje wszystkie mozliwe
 * -- rozwiazania, dla kazdego z nich wartosc predykatu jest false wiec zmienia je na nowe, ostatecznie i tak
 * -- wartosc jest false
 */
/* -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  -- */


/* -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  -- */
/* Zadanie 3 */

% From the book
% PROLOG PROGRAMMING IN DEPTH
% by Michael A. Covington, Donald Nute, and Andre Vellino
% (Prentice Hall, 1997).
% Copyright 1997 Prentice-Hall, Inc.
% For educational use only

% File CAPITALS.PL or KB.PL
```

```
% Knowledge base for several examples in Chapter 2

:- dynamic(capital_of/2).      % Remove if not needed. See text, section 2.8.

capital_of(georgia,atlanta).
capital_of(california,sacramento).
capital_of(florida,tallahassee).
capital_of(maine,augusta).

 % From the book
% PROLOG PROGRAMMING IN DEPTH
% by Michael A. Covington, Donald Nute, and Andre Vellino
% (Prentice Hall, 1997).
% Copyright 1997 Prentice-Hall, Inc.
% For educational use only

% File LEARNER.PL
% Program that modifies its own knowledge base

%  This program requires file KB.PL, which should be a copy of CAPITALS.PL.

start :-  consult('learner_kb.pl'),
          nl,
          write('Type names entirely in lower case, followed by period.'), nl,
          write('Type "stop." to quit.'), nl,
          nl,
          process_a_query.

process_a_query :- write('State? '),
                   read(State),
                   answer(State).

   % If user typed "stop." then save the knowledge base and quit.

answer(stop) :-    write('Saving the knowledge base...'),nl,
                   tell('learner_kb.pl'),
                   write(':- dynamic(capital_of/2).'),nl,   % omit if not needed
                   listing(capital_of),
                   told,
                   write('Done.'),nl.

   % If the state is in the knowledge base, display it, then
   % loop back to process_a_query

answer(State) :-   capital_of(State,City),
                   write('The capital of '),
                   write(State),
                   write(' is '),
                   write(City),nl,
                   nl,
                   process_a_query.

   % If the state is not in the knowledge base, ask the
   % user for information, add it to the knowledge base, and
   % loop back to process_a_query

answer(State) :-   \+ capital_of(State,_),
                   write('I do not know the capital of that state.'),nl,
                   write('Please tell me.'),nl,
                   write('Capital? '),
                   read(City),
                   write('Thank you.'),nl,nl,
                   assertz(capital_of(State,City)),
                   process_a_query.


/* Jakie 3 przypadki odpowiedzi są brane pod uwagę?
* 1. stop - zapis wiedzy do pliku 2. - wpisanie stanu ktory jest w bazie wiedzy -> wypisanie stolicy
* 3. wpisanie stanu ktorego nie ma w bazie wiedzy -> dodanie stanu i stolicy do bazy wiedzy
* Co dzieje się przy wyjściu z programu i jak to wpływa na jego kolejne uruchamianie?
* Przy wyjsciu z programu zapisywana jest aktualna wiedza, przy nastepnym uruchomieniu
* wiedza ta jest wiedza poczatkowa
*/
/* -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  -- */


/* Zadanie 4 */
/* -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  -- */
delta(A,B,C,Wynik) :-
    Wynik is B * B - 4 * A * C.

kwadrat(A,B,C,Wynik) :-
    delta(A,B,C,Del),
    Del < 0,
    write('Delta less than zero'),
    nl,
    write('No solutions'),
    Wynik is -1,
    fail.

kwadrat(A,B,C,Wynik) :-
    delta(A,B,C,Del),
```

```
    delta(A,B,C,Del),
    Del =:= 0,
    write('Delta equals 0, One solution'),
    Wynik is -1 * B / 2  * A.

kwadrat(A,B,C,Wynik) :-
    delta(A,B,C,Del),
    Del > 0,
    write('Two solutions'),
    nl,
    Wynik is ( -B - sqrt(Del) ) / 2 * A,
    nl.

kwadrat(A,B,C,Wynik) :-
    delta(A,B,C,Del),
    Del > 0,
    write('Two solutions'),
    nl,
    Wynik is ( -B + sqrt(Del) ) / 2 * A,
    nl.
/* -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  -- */

/* Zadanie 5 */
/* -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  -- */
factorial(0,1).

factorial(Number,Result) :-
    Number > 0,
    NewNumber is Number - 1,
    factorial(NewNumber, NewResult),
    Result is Number * NewResult.

fibonacci(0,0).
fibonacci(1,1).
fibonacci(N,Result) :-
    Num is N,
    X1 is Num-1,
    Y1 is Num-2,
    fibonacci(X1,X),
    fibonacci(Y1,Y),
    Result is X+Y.
/* -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --  -- */
```