```
% Problem 1: Pakde, Bude, Paklik, and Bulik
% parent(X,Y) denotes that X is one of Y's parent
parent(anya,clara). parent(anya,edward). parent(anya,fiona).
parent(benjamin,clara). parent(benjamin,edward). parent(benjamin,fiona).

parent(clara,hannah). parent(clara,ira). parent(clara,lucas).
parent(david,hannah). parent(david,ira). parent(david,lucas).

parent(fiona,mike). parent(fiona,nancy).
parent(george,mike). parent(george,nancy).

parent(ira,peter). parent(ira,quincy).
parent(james,peter). parent(james,quincy).

parent(kiana,raymond). parent(kiana,sarah). parent(kiana,tina).
parent(lucas,raymond). parent(lucas,sarah). parent(lucas,tina).

parent(nancy,umberto). parent(nancy,victoria).
parent(oscar,umberto). parent(oscar,victoria).

/* person(Person,Gender,Birthyear) explains that the
person Person is of gender Gender and born at Birthyear */

person(anya,female,1938). person(benjamin,male,1929).

person(clara,female,1959). person(david,male,1950).
person(edward,male,1963).
person(fiona,female,1965). person(george,male,1955).

person(hannah,female,1980).
person(ira,female,1982). person(james,male,1979).
person(kiana,female,1990). person(lucas,male,1989).

person(mike,male,1991).
person(nancy,female,1994). person(oscar,male,1992).

person(peter,male,2005). person(quincy,female,2008).
person(raymond,male,2013). person(sarah,female,2015). person(tina,female,2018).

person(umberto,male,2016). person(victoria,female,2019).

% male(X): X is male. female(X): X is female
male(X):- person(X,male,_).
female(X):- person(X,female,_).

/*some rules and predicates from PS 1 and Hw 1*/
% married(X,Y): X is married to Y (or vice versa)
married(X,Y):- parent(X,Z),parent(Y,Z), X \== Y.

% sibling(X,Y): X is sibling of Y
% brother(X,Y): X is a brother of Y. sister(X,Y): X is a sister of Y.
sibling(X,Y):- parent(Z,X), parent(Z,Y), X \== Y.
brother(X,Y):- sibling(X,Y), male(X).
```

```
sister(X,Y):- sibling(X,Y), female(X).

% older(X,Y) means X born before Y
% younger(X,Y) means X born after Y
older(X,Y):- person(X,_,YearX), person(Y,_,YearY), YearX < YearY.
younger(X,Y):- person(X,_,YearX), person(Y,_,YearY), YearX > YearY.

% older_brother(X,Y): X is an older brother of Y
older_brother(X,Y):- brother(X,Y), older(X,Y).
% younger_brother(X,Y): X is a younger brother of Y
younger_brother(X,Y):- brother(X,Y), younger(X,Y).
% older_sister(X,Y): X is an older sister of Y
older_sister(X,Y):- sister(X,Y), older(X,Y).
% younger_sister(X,Y): X is a younger sister of Y
younger_sister(X,Y):- sister(X,Y), younger(X,Y).

% pakde(X,Y): X is a pakde of Y
pakde(X,Y):- older_brother(X,Z), parent(Z,Y). % pakde by blood
pakde(X,Y):- married(X,Z), older_sister(Z,W), parent(W,Y). % pakde by marriage

% bude(X,Y):- X is a bude of Y
bude(X,Y):- older_sister(X,Z), parent(Z,Y). % bude by blood
bude(X,Y):- married(X,Z), older_brother(Z,W), parent(W,Y). % bude by marrige

% paklik(X,Y): X is a paklik of Y
paklik(X,Y):- younger_brother(X,Z), parent(Z,Y). % paklik by blood
paklik(X,Y):- married(X,Z), younger_sister(Z,W), parent(W,Y). % paklik by marriage

% bulik(X,Y):- X is a bulik of Y
bulik(X,Y):- younger_sister(X,Z), parent(Z,Y). % bulik by blood
bulik(X,Y):- married(X,Z), younger_brother(Z,W), parent(W,Y). % bulik by marriage

% END OF PROBLEM 1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Problem 2: Perjalanan
:- op(600, xfx, langsung_ke).

jakarta langsung_ke bekasi. jakarta langsung_ke bogor.

bogor langsung_ke bandung. bogor langsung_ke sukabumi.

bekasi langsung_ke bandung. bekasi langsung_ke cirebon.

bandung langsung_ke cirebon. bandung langsung_ke tasikmalaya.

cirebon langsung_ke tasikmalaya. cirebon langsung_ke tegal.

tegal langsung_ke pekalongan.

pekalongan langsung_ke semarang. pekalongan langsung_ke yogyakarta.

semarang langsung_ke surakarta. semarang langsung_ke surabaya.

yogyakarta langsung_ke semarang. yogyakarta langsung_ke surakarta.

surakarta langsung_ke madiun.

madiun langsung_ke kediri. madiun langsung_ke surabaya.

kediri langsung_ke malang.

surabaya langsung_ke malang.

:- op(600, xfx, ke).
X ke Y:- X langsung_ke Y.
X ke Y:- X langsung_ke Z, Z ke Y.

:- op(600, xfx, menuju).
X menuju Y:- X langsung_ke Y.
X menuju Y:-
    Z langsung_ke Y,
    X menuju Z, write(Z), write(' ').

% END OF PROBLEM 2 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Problem 3: Triangle
% star(N) generates a string of N stars, every two stars are separated by a space
star(1):- write('*'). % base case
star(N):- N > 1, write(*),write(' '), N1 is N - 1, star(N1). % recursive case

% space(N) generates a string of N blank spaces
space(0):- write('').
space(N):- N > 0, write(' '), N1 is N-1, space(N1).

/* triangle(N,MaxStars) generates a string of MaxStars lines of stars,
each line N consists of N stars preceded by MaxStars - N - 1 blank spaces */
triangle(N,MaxStars):-
    Space is N - 1, Star is MaxStars - Space,
    space(Space),star(Star),nl,
    triangle(Space,MaxStars).

/* triangle(N) generates N lines of string, for each 1 =< i =< N, line i
contains i stars, every two stars is separated by a space, every first star
of line i is preceded by N - i - 1 blank spaces */
triangle(N):- N > 0, triangle(N,N).

% END OF PROBLEM 3 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Problem 4: Fibonacci Sequence
/* the predicate fibo(+F0,+F1,N,FN) returns the N-th
fibonacci sequence, i.e., FN, with initial condition F0 and F1,
the sequence is defined recursively: F(N) = F(N-1) + F(N-2) for N > 1 */

% base cases:
fibo(F0,_,0,F0).
fibo(_,F1,1,F1).
% recursive case:
fibo(F0,F1,N,FiboN):-
    N > 1,
    N1 is N - 1, N2 is N - 2,
    fibo(F0,F1,N1,FiboN1), fibo(F0,F1,N2,FiboN2), % calling fibo/4 recursively
    FiboN is FiboN1 + FiboN2.

% END OF PROBLEM 4 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Problem 5: Genap and Ganjil
:- op(700,xf, genap).
A genap:- A mod 2 =:= 0.

:- op(700,xf, ganjil).
A ganjil:- A mod 2 =\= 0.
% END OF PROBLEM 5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
% Problem 6: Primitive Pythagorean Triangle

% the predicate from Problem 5 in Hw 3, version 2
gcd(0,0,_):- write("gcd error"). % handling exception when both numbers are zero
gcd(A,0,A):- A =\=0. % gcd(A,0) = A for nonzero A
gcd(0,A,A):- A =\=0. % gcd(0,A) = A for nonzero A
gcd(A,A,A):- A =\=0. % gcd(A,A) = A for nonzero A
gcd(A,B,Gcd):-  % if 0 < A < B, then gcd(A,B) = gcd (A,B-A) (3rd case).
    0 < A,
    A < B,
    C is B-A,
    gcd(A,C,Gcd).
gcd(A,B,Gcd):- % If 0 < B < A, then gcd(A,B) = gcd(A-B,B) (4th case).
    0 < B,
    B < A,
    gcd(B,A,Gcd).

/* gcd(A,B,C,Gcd) holds whenever Gcd is the greatest common divisor of A, B, and C,
to calculate Gcd, we use the following theorem: gcd(A,B,C) = gcd(gcd(A,B),C) */
gcd(A,B,C,Gcd):- gcd(A,B,GcdAB),gcd(GcdAB,C,Gcd).

/* primtriple(A,B,C) succeeds whenever (A,B,C) is a primitive Pythagorean triple,
i.e., A < B < C, A^2 + B^2 = C^2 and gcd(A,B,C) = gcd(A,gcd(B,C)) = 1 */

primtriple(A,B,C):-
    integer(A),integer(B),integer(C),
    0 < A, A < B, B < C,
    A^2 + B^2 =:= C^2,
    gcd(A,B,C,1).

/* ptriangle(A,B,C,Perimeter) succeeds whenever (A,B,C) is a Pythagorean triple and
A + B + C = Perimeter, i.e., A, B, C are the sides of a primitive Pythagorean triangle
with 0< A < B < C whose perimeter is equal to Perimeter */
ptriangle(A,B,C,Perimeter):-
    integer(Perimeter), Perimeter > 0,
    HalfPerimeter is Perimeter // 2,
    between(1,HalfPerimeter,A),between(1,HalfPerimeter,B),
    C is Perimeter - A - B, % since A + B + C = Perimeter
    primtriple(A,B,C).

ppt((A,B,C),Min,Max):- between(Min,Max,Perimeter), ptriangle(A,B,C,Perimeter).
```