BAGIAN I : Fakta, Rule, dan Query

1. Deklarasi Fakta

```
pria(yoga).
pria(zayn_malik).
pria(padil).
pria(jovan).
pria(zunan).
pria(farras).
pria(william).
pria(small_faris).
pria(baby_thajeb).
wanita(lisa).
wanita(asin).
wanita(rikha).
wanita(siti).
wanita(nurbaya).

usia(yoga,71).
usia(lisa,65).
usia(zayn_malik,56).
usia(asin,51).
usia(padil,58).
usia(rikha,40).
usia(jovan,24).
usia(zunan,30).
usia(farras,32).
usia(siti,26).
usia(william,28).
usia(nurbaya,24).
usia(small_faris,6).
usia(baby_thajeb,3).

menikah(yoga,lisa).
menikah(lisa,yoga).
menikah(zayn_malik,asin).
menikah(asin,zayn_malik).
menikah(padil,rikha).
menikah(rikha,padil).
menikah(farras,siti).
menikah(siti,farras).
menikah(william,nurbaya).
menikah(nurbaya,william).

anak(jovan,yoga).
anak(jovan,lisa).
anak(zunan,yoga).
anak(zunan,lisa).
anak(farras,yoga).
anak(farras,lisa).
anak(siti,zayn_malik).
anak(siti,asin).
anak(william,zayn_malik).
anak(william,asin).
anak(nurbaya,padil).
anak(nurbaya,rikha).
anak(small_faris,farras).
anak(small_faris,siti).
anak(baby_thajeb,william).
anak(baby_thajeb,nurbaya).

saudara(jovan,zunan).
saudara(jovan,farras).
saudara(zunan,jovan).
saudara(zunan,farras).
saudara(farras,jovan).
saudara(farras,zunan).
saudara(siti,william).
saudara(william,siti).
```

2. Deklarasi Rules

```
kakak(X,Y) :-
    saudara(X,Y),
    usia(X,U),
    usia(Y,V),
    U>V.

keponakan(X,Y) :-
    anak(X,U),
    saudara(U,Y).

suami(X,Y) :-
    menikah(X,Y),
    pria(X).

sepupu(X,Y) :-
    anak(X,U),
    saudara(U,V),
    anak(Y,V).

mertua(X,Y) :-
```

```
    menikah(Y,U),
    anak(U,X).

bibi(X,Y) :-
    saudara(X,U),
    anak(Y,U),
    wanita(X).

cucu(X,Y) :-
    anak(X,U),
    anak(U,Y).

anaksulung(X) :-
    anak(X,Y),
    wanita(Y), /*atau pria(Y)*/
    \+kakak(_,X).

anakbungsu(X) :-
    anak(X,Y),
    wanita(Y), /*atau pria(Y)*/
    \+kakak(X,_).
```

3. Query

a. Suami dari Nurbaya

```
| ?- suami(X,nurbaya).

X = william ?

yes
```

b. Paman dari Small Faris

```
| ?- keponakan(small_faris,X), pria(X).

X = jovan ? ;

X = zunan ? ;

X = william

(16 ms) yes
```

c. Menantu dari Yoga

```
| ?- mertua(yoga,Y).

Y = siti ?
```

```
(16 ms) yes
```

d. Nenek dari Small Faris

```
| ?- cucu(small_faris,X), wanita(X).

X = lisa ? ;

X = asin

yes
```

e. Cucu dari Padil

```
| ?- cucu(X,padil).

X = baby_thajeb ?

yes
```

f. Ipar dari Siti

```
| ?- menikah(siti,X), saudara(X,Y).

X = farras
Y = jovan ? ;

X = farras
Y = zunan

yes
```

g. Sepupu dari Baby Thajeb

```
| ?- sepupu(baby_thajeb,X).

X = small_faris ?

(16 ms) yes
```

h. Wanita yang merupakan anak tunggal

```
| ?- wanita(X), anak(X,_), \+saudara(X,_).

X = nurbaya ?

yes
```

i. Pria yang belum menikah

```
| ?- pria(X), \+menikah(X,_).
```

```
X = jovan ? ;
X = zunan ? ;
X = small_faris ? ;
X = baby_thajeb

(47 ms) yes
```

BAGIAN II : Rekurens

faktorial(N,X)

```
faktorial(0, X, X) :- !.

faktorial(N, Current, X) :-
    NewN is N-1,
    NewCurrent is Current * N,
    faktorial(NewN, NewCurrent, X).

faktorial(N,X) :- faktorial(N, 1, X).
```

alternatif :

```
faktorial(0,1) :- !.
faktorial(N,X) :- N1 is N-1,
                faktorial(N1,X1),
                X is N*X1.
```

gcd(A,B,X)

(djikstra)

```
gcd(0, X, X) :-!.
gcd(X, 0, X) :-!.
gcd(X, X, X) :-!.

gcd(M, N, X) :-
    N>M,
    Y is N-M,
    gcd(M, Y, X).

gcd(M, N, X) :-
    N<M,
    Y is M-N,
    gcd(Y, N, X).
```

alternatif :

(euclidean)

```
gcd(X,0,X) :- !.
gcd(0,X,X) :- !.
gcd(A,B,X) :- A1 is A mod B,
            gcd(B,A1,X).
```

power(A,B,X)

```
power(_, -1, Current, Current) :- !.
power(_, 0, _, 1) :- !.
power(_, 1, Current, Current) :- !.

power(A, B, Current, X) :-
    B > 0,
    NewCurrent is Current * A,
    NewB is B-1,
    power(A, NewB, NewCurrent, X).

power(A, B, Current, X) :-
    B < 0,
    NewCurrent is Current / A,
    NewB is B+1,
    power(A, NewB, NewCurrent, X).

power(A, B, X) :- B > 0, power(A, B, A, X).
power(A, B, X) :- B < 0, NewA is 1/A, power(A, B, NewA, X).
power(A, B, X) :- A \= 0, B = 0, X is 1.
```

alternatif :

```
power(A,0,1) :- A\=0,!.
power(A,B,X) :- B>0, B1 is B-1, power(A,B1,X1), X is A*X1.
power(A,B,X) :- B<0, A1 is 1/A, B1 is B*(-1), power(A1,B1,X).
```

countDigit(A,X)

```
countDigit(0,1,0) :- !.
countDigit(0,Count,Count) :- !.
countDigit(A,X,Count) :-
    NewA is A//10,
    NewCount is Count + 1,
    countDigit(NewA,X,NewCount).
countDigit(A,X) :- countDigit(A,X,0).
```

alternatif :

```
countDigit(A,X) :- A<10, X is 1,!.
countDigit(A,X) :- A1 is A div 10, countDigit(A1,X1), X is
X1+1.
```

createTriangle(X)

```
star(B,B) :- !.
star(Current,B) :-
    write('*'),
    NewCurrent is Current+1,
    star(NewCurrent,B).

createTriangle(_,0) :- !.
createTriangle(X,Current) :-
    star(0,Current),
    write('\n'),
    NewCurrent is Current - 1,
    createTriangle(X,NewCurrent).

createTriangle(X) :- createTriangle(X,X).
```

alternatif:

```
printLine(0) :- nl,!.
printLine(Y) :-
    write('*'),
    Y1 is Y-1,
    printLine(Y1).

createTriangle(0) :- !.
createTriangle(X) :-
    X>0,
    printLine(X),
    X1 is X-1,
    createTriangle(X1).
```

BAGIAN III : List

push(Element, Queue, Result)

```
push(X, [], [X]).
push(Z, [X|Y], [X|W]) :- push(Z,Y,W).
```

Simulasi push :

```
push(3, [1,2], Result)
push(3, 1 | [2], 1 | W) :- push(3, [2], W)
push(3, 2 | [], 2 | W) :- push(3, [], W)
push(3,[],W) -> basis -> W = [3]

W naik lagi sampai Result, hasil [1,2,3]
```

pop(Queue, Result)

```
pop([_|X], X).
```

front(Queue, Result)

```
front([X|_], X).
```

back(Queue, Result)

```
back([X], X).
back([_|Z], X) :- back(Z, X).
```

concatenate(FList, SList, X, Y, Result)

```
add(_, 0, []) :- !.
add([X|Y], Count, [X|W]) :-
    NewCount is Count-1,
    add(Y, NewCount, W).

concatList([], X, X).
concatList([X|Y], Z, [X|W]) :- concatList(Y, Z, W).

concatenate(FList, SList, X, Y, Result) :-
    add(FList, X, Result1),
    add(SList, Y, Result2),
    concatList(Result1, Result2, Result).
```

palindrom(List)

```
reverseList(X,Y) :- reverseList(X,[],Y).
reverseList([],X,X).
reverseList([X|Y],Z,T) :- reverseList(Y,[X|Z],T).

palindrom(X) :-
    reverseList(X,NewList),
    X = NewList.
```

Bonus

ridge(List)

```
:- dynamic(down/1).
:- dynamic(count/1).

/* When Go Up */
ridge([X|Y], Before) :-
    down(_),
    count(L),
    Before < X,
    NewL is L+1,
    retractall(count(_)),
    retractall(down(_)),
    asserta(count(NewL)),
    ridge(Y,X),!.

ridge([X|Y], Before) :-
    \+down(_),
    Before < X,
    ridge(Y, X),!.

/* When Go Down */
ridge([X|Y], Before) :-
    Before >= X,
    asserta(down(1)),
    ridge(Y, X),!.

ridge([], _) :-
    count(L),
    L \= 0,
    write(L),!.

ridge([X|Y]) :-
    retractall(count(_)),
    retractall(down(_)),
    asserta(count(0)),
    ridge(Y, X),!.
```