28.Write a Prolog Program for forward Chaining. Incorporate required queries.

% Declare facts as dynamic to allow adding during runtime

:- dynamic known/1.

% Rules (IF conditions THEN conclusion)

rule(headache, fever, flu).

rule(fever, cough, covid).

rule(cough, sore\_throat, cold).

rule(cold, sneezing, allergy).

rule(body\_ache, flu, dengue).

% Starting known facts (can also be empty)

fact(fever).

fact(cough).

fact(sore\_throat).

% Forward chaining engine

forward :-

retractall(known(\_)), % clear known facts

findall(F, fact(F), Facts),

assert\_facts(Facts),

infer\_new\_facts,

nl, write('Derived Facts:'), nl,

list\_known\_facts.

% Assert all starting facts

assert\_facts([]).

assert\_facts([H|T]) :-

assertz(known(H)),

assert\_facts(T).

% Inference loop: try to infer until no new facts found

infer\_new\_facts :-

( infer\_one -> infer\_new\_facts ; true ).

% Apply one rule if possible

infer\_one :-

rule(A, B, C),

known(A),

known(B),

\+ known(C),

assertz(known(C)),

format('Inferred: ~w from ~w and ~w~n', [C, A, B]).

% Display all known facts

list\_known\_facts :-

known(F),

format('- ~w~n', [F]),

fail.

list\_known\_facts.

OUTPUT

