**1) BACKWARD CHAIN**

parent(john, mary).

parent(mary, susan).

parent(susan, alice).

ancestor(X, Y) :- parent(X, Y). % Direct parent

ancestor(X, Y) :- parent(X, Z), ancestor(Z, Y). % Recursive rule

**2) BFS**

edge(a, b).

edge(a, c).

edge(b, d).

edge(c, d).

path(X, Y, [X, Y]) :- edge(X, Y).

path(X, Y, [X|Path]) :- edge(X, Z), path(Z, Y, Path).

**3. BIRD**

bird(eagle,can\_fly).

bird(sparrow,can\_fly).

bird(penguin,cannot\_fly).

can\_fly(Bird):-bird(Bird,can\_fly).

bird(Bird,can\_fly).

bird(Bird,cannot\_fly).

**4) DISEASE**

% Facts: Diseases and their recommended diets

diet(diabetes, [low\_sugar, high\_fiber, low\_carb]).

diet(hypertension, [low\_sodium, high\_potassium, low\_fat]).

diet(obesity, [low\_calorie, high\_protein, high\_fiber]).

diet(cholesterol, [low\_cholesterol, high\_fiber, low\_saturated\_fat]).

% Rule: Suggest diet based on disease

suggest\_diet(Disease, Diets) :-

diet(Disease, Diets).

% Example query:

% ?- suggest\_diet(diabetes, Diets).

**5) DOB**

dob(uday, date(2000, 4, 10)).

dob(john, date(1995, 6, 15)).

**6) FAMILY TREE**

father(john, alice).

father(john, bob).

mother(mary, alice).

mother(mary, bob).

husband(john, mary).

wife(mary, john).

parent(X, Y) :- father(X, Y).

parent(X, Y) :- mother(X, Y).

sibling(X, Y) :- parent(Z, X), parent(Z, Y), X \= Y.

**7) FORWARD CHAIN**

fact(smoke\_detected).

fire\_alarm :- fact(smoke\_detected), assert(fact(fire\_alarm)).

activate\_sprinkler :- fact(fire\_alarm), write('Sprinkler activated.').

run\_forward\_chain :-

fire\_alarm,

activate\_sprinkler.

**8) FRUIT AND COLOUR**

fruit(apple, red).

fruit(banana, yellow).

fruit(cherry, red)edge(a, b).

edge(a, c).

edge(b, d).

edge(c, d).

path(X, Y, [X, Y]) :- edge(X, Y).

path(X, Y, [X|Path]) :- edge(X, Z), path(Z, Y, Path).

**9) MEDICAL DIAGNOSIS**

% Facts: Diseases and their recommended diets

diet(diabetes, [low\_sugar, high\_fiber, low\_carb]).

diet(hypertension, [low\_sodium, high\_potassium, low\_fat]).

diet(obesity, [low\_calorie, high\_protein, high\_fiber]).

diet(cholesterol, [low\_cholesterol, high\_fiber, low\_saturated\_fat]).

% Rule: Suggest diet based on disease

suggest\_diet(Disease, Diets) :-

diet(Disease, Diets).

% Example query:

% ?- suggest\_diet(diabetes, Diets).

**10) MONKEY BANANA**

initial(state(monkey, box, bananas, on\_ground)).

goal(state(monkey, box, bananas, holding)).

action(push\_box, state(monkey, box, bananas, on\_ground), state(monkey, on\_box, bananas, on\_ground)).

action(climb\_box, state(monkey, on\_box, bananas, on\_ground), state(monkey, on\_box, bananas, on\_box)).

action(reach\_bananas, state(monkey, on\_box, bananas, on\_box), state(monkey, on\_box, bananas, holding)).

plan(State, State, []).

plan(State, Goal, [Action | Rest]) :-

action(Action, State, NewState),

plan(NewState, Goal, Rest).

find\_plan(Plan) :-

initial(InitialState),

goal(GoalState),

plan(InitialState, GoalState, Plan).

**12)STUDENT TEACHER**

teaches(mr\_smith, john).

teaches(mr\_smith, alice).

teaches(ms\_jones, bob).

teaches(ms\_jones, emily).

student(X) :- teaches(\_, X).

teacher(Y) :- teaches(Y, \_).

students\_of(Teacher, Students) :- findall(Student, teaches(Teacher, Student), Students).

**13) TOWERS OF HANOI**

move(1, A, B, \_) :-

write('Move disk from '), write(A), write(' to '), write(B), nl.

move(N, A, B, C) :-

N > 1,

M is N - 1,

move(M, A, C, B),

move(1, A, B, \_),

move(M, C, B, A).

hanoi(N) :-

move(N, 'A', 'B', 'C').

**14) SUM 1 TO N**

sum(0,0).

sum(N,S):-N>0, N1 is N-1,

sum(N1,S1),S is S1+N.

sum(5,S).

**15) PERSON**

person('teju','6789').

person('name').

query(Person):-person(Person),writeln(Person),fail.

**16) PLANET DB**

planet(mercury, terrestrial).

planet(earth, terrestrial).

planet\_type(Planet, Type) :-

planet(Planet, Type).

planet\_name(Planet) :-

planet(Planet, \_).