Prolog – Monkey and Banana Problem

```
at(monkey, door).

at(box, middle).

at(banana, ceiling).

onfloor(monkey).

onfloor(box).

hungry(monkey).

walk(monkey, middle) :- at(monkey, door).

push(box, middle) :- at(box, middle).

climb(monkey, box) :- onfloor(monkey), at(monkey, middle).

grasp(banana) :- climb(monkey, box), at(banana, ceiling), write('Monkey got the banana!').
```

Prolog – Object Location

```
location(spoon, kitchen).

location(phone, hall).

location(book, study).

find(Item) :- location(Item, Place),

        write(Item), write(' is in '), write(Place), nl.

% Query: find(spoon).
```

Prolog – Mortal Check

```
man(socrates).

man(plato).

man(aristotle).

mortal(X) :- man(X).

% Query: mortal(socrates).
```

Prolog – Count Vowels in a Sentence

```
sentence("I am an intelligent AI model").

vowel(a). vowel(e). vowel(i). vowel(o). vowel(u).

count_vowels(Count) :-

  sentence(S),

  string_chars(S, Chars),

  include(vowel, Chars, Vowels),
    length(Vowels, Count).

% Query: count_vowels(C).
```

Prolog – Forward Chaining (Alice is a criminal)

```
% Facts

stole(alice, painting).

property(painting, bob).

% Rules

illegal(X) :- stole(X, Y), property(Y, _).

criminal(X) :- illegal(X).

% Query: criminal(alice).
```

Prolog – Can Bird Swim or Not

```
bird(eagle).

bird(sparrow).

bird(penguin).

bird(duck).

swim(penguin) :- !, fail.

swim(_) :- write('Bird can swim').
```

Prolog – Animal Classification

```
% Bird if has feathers and lays eggs

% Mammal if has fur and gives birth

% Reptile if has scales and lays eggs

has_feathers(eagle).

lays_eggs(eagle).

has_fur(dog).

gives_birth(dog).

has_scales(crocodile).

lays_eggs(crocodile).

type(X, bird) :- has_feathers(X), lays_eggs(X).

type(X, mammal) :- has_fur(X), gives_birth(X).

type(X, reptile) :- has_scales(X), lays_eggs(X).

% Query example: type(eagle, T).
```

Prolog – Vacuum Cleaner Problem

```prolog
% Vacuum World Problem

dirty(a).

dirty(b).

clean(X) :- \+ dirty(X), write(X), write(' is already clean'), nl.

clean(X) :- dirty(X), retract(dirty(X)), write('Cleaning '), write(X), nl.

start :-

   (clean(a); true),

   (clean(b); true),

   write('Done.').
```

Prolog – Family Tree

```prolog
female(pam). female(liz). female(ann). female(pat).

male(tom). male(bob). male(jim).

mother(pam, liz).

father(tom, bob).

father(tom, liz).

father(bob, jim).

mother(ann, jim).

grandfather(X,Y) :- father(X,Z), father(Z,Y).

grandmother(X,Y) :- mother(X,Z), mother(Z,Y).

sister(X,Y) :- female(X), father(F, X), father(F, Y), X \= Y.

brother(X,Y) :- male(X), father(F, X), father(F, Y), X \= Y.
```

Prolog – Umbrella and Weather

```prolog
rainy(mumbai).

rainy(delhi).

windy(chennai).

cold(delhi).

carry_umbrella(X) :- rainy(X).

carry_umbrella(X) :- windy(X).

warm_clothes(X) :- cold(X), rainy(X).

% Example Queries:
```

```
% carry_umbrella(mumbai).

% warm_clothes(delhi).
```

Prolog – Vehicle Type and Fuel

```
type(car, four_wheeler).

type(bike, two_wheeler).

type(bus, four_wheeler).

type(scooter, two_wheeler).

fuel(car, petrol).

fuel(bike, petrol).

fuel(bus, diesel).

fuel(scooter, petrol).

vehicle(X, Type, Fuel) :- type(X, Type), fuel(X, Fuel).

% Query: vehicle(car, Type, Fuel).
```

Prolog – Book and Author Matching

```
book_author(the_hobbit, tolkien).

book_author(the_fellowship_of_the_ring, tolkien).

book_author(harry_potter, rowling).

book_author(chamber_of_secrets, rowling).

book_genre(the_hobbit, fantasy).

book_genre(the_fellowship_of_the_ring, fantasy).

book_genre(harry_potter, fantasy).

book_genre(chamber_of_secrets, fantasy).

% Queries:

% book_author(Book, tolkien).

% book_genre(the_hobbit, Genre).
```

Prolog – Royal Family Tree

```
king(george_v).

queen(elizabeth).

prince(charles).

prince(william).

princess(diana).
```

```prolog
parent(george_v, elizabeth).
parent(elizabeth, charles).
parent(charles, william).
spouse(charles, diana).
spouse(william, kate).
grandparent(X,Y) :- parent(X,Z), parent(Z,Y).
child(Y,X) :- parent(X,Y).
sibling(X,Y) :- parent(Z,X), parent(Z,Y), X \= Y.
% Query: grandparent(GP, william).
```

Prolog – Inference Engine

```prolog
rule(likes_milk, (mammal(X), not(carnivore(X)))).
rule(can_fly, (bird(X), not(penguin(X)))).
mammal(cow).
mammal(elephant).
carnivore(tiger).
bird(eagle).
bird(penguin).
% Queries:
% rule(likes_milk, Condition), call(Condition).
% rule(can_fly, Condition), call(Condition).
```

Prolog – Car Fault Diagnosis

```prolog
problem(car1, engine_noise).
problem(car1, oil_leak).
problem(car2, battery_low).
problem(car2, slow_start).
problem(car3, flat_tire).
diagnosis(car1, engine_failure) :- problem(car1, engine_noise), problem(car1, oil_leak).
diagnosis(car2, battery_issue) :- problem(car2, battery_low), problem(car2, slow_start).
diagnosis(car3, tire_puncture) :- problem(car3, flat_tire).
% Query: diagnosis(car1, Fault).
```

Prolog – Plant Classification

```
feature(rose, flowering).

feature(rose, thorny).

feature(cactus, succulent).

feature(cactus, thorny).

feature(mango, fruit_bearing).

feature(mango, woody).

classify(rose, shrub) :- feature(rose, flowering), feature(rose, thorny).

classify(cactus, succulent) :- feature(cactus, succulent).

classify(mango, tree) :- feature(mango, fruit_bearing), feature(mango, woody).

% Queries:

% classify(rose, Type).

% classify(mango, Type).
```

Prolog – Weather Prediction

```
symptom(chennai, humid).

symptom(chennai, cloudy).

symptom(delhi, sunny).

symptom(delhi, dry).

symptom(ooty, cold).

symptom(ooty, foggy).

hypothesis(chennai, rainy) :- symptom(chennai, humid), symptom(chennai, cloudy).

hypothesis(delhi, clear) :- symptom(delhi, sunny), symptom(delhi, dry).

hypothesis(ooty, foggy_weather) :- symptom(ooty, cold), symptom(ooty, foggy).

% Query: hypothesis(chennai, Weather).
```

Prolog – Student Academic Performance

```
attribute(john, hardworking).

attribute(john, regular).

attribute(sarah, irregular).

attribute(sarah, average).

attribute(mike, hardworking).

attribute(mike, irregular).
```

```
performance(Student, excellent) :- attribute(Student, hardworking), attribute(Student, regular).

performance(Student, good) :- attribute(Student, hardworking), attribute(Student, irregular).

performance(Student, average) :- attribute(Student, average).

% Query: performance(john, Level).
```

Prolog – Student, Teacher, and Subject Matching

```
studies(charlie, csc135).

studies(olivia, csc135).

studies(jack, csc131).

studies(arthur, csc134).

teaches(kirke, csc135).

teaches(collins, csc131).

teaches(collins, csc171).

teaches(juniper, csc134).

professor(X, Y) :- teaches(X, C), studies(Y, C).

% Query: professor(kirke, Student).
```

Prolog – 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).

% Query: move(3, left, right, center).
```

Prolog – Forward Chaining: Robert is Criminal

```
% Facts

american(robert).

weapon(missile).

sells(robert, missile, country_a).

enemy(country_a, america).
```

```
% Rules

criminal(X) :- american(X), sells(X, Y, Z), weapon(Y), enemy(Z, america).

% Query: criminal(robert).
```

Prolog – Dog and Cat Facts

```
dog(fido).

dog(rover).

dog(jane).

dog(tom).

dog(fred).

dog(henry).

cat(mary).

cat(harry).

cat(bill).

cat(steve).

small(henry).

medium(harry).

medium(fred).

large(fido).

large(mary).

large(tom).

large(fred).

large(steve).

large(jim).

large(mike).

% Example Query: large(X).
```

Prolog – Planets Database

```
orbits(mercury, sun).

orbits(venus, sun).

orbits(earth, sun).

orbits(mars, sun).

orbits(moon, earth).
orbits(phobos, mars).

orbits(deimos, mars).

% Query: orbits(X, sun).
```

Prolog – Forward Chaining: Rain and Cold

```
rainy(chennai).

rainy(coimbatore).

rainy(ooty).

cold(ooty).

carry_umbrella(X) :- rainy(X).

wear_jacket(X) :- rainy(X), cold(X).

% Queries:

% carry_umbrella(chennai).

% wear_jacket(ooty).
```

Prolog – Fruit Color with Backtracking

```
colour(cherry, red).

colour(banana, yellow).

colour(apple, red).

colour(apple, green).

colour(orange, orange).

colour(X, unknown).

% Queries:

% colour(apple, Color).
```

Prolog – Pattern Matching

```
first_name(tonyblair, tony).

first_name(georgebush, georgedubya).

second_name(tonyblair, blair).

second_name(georgebush, bush).

% Query: first_name(X, tony).
```