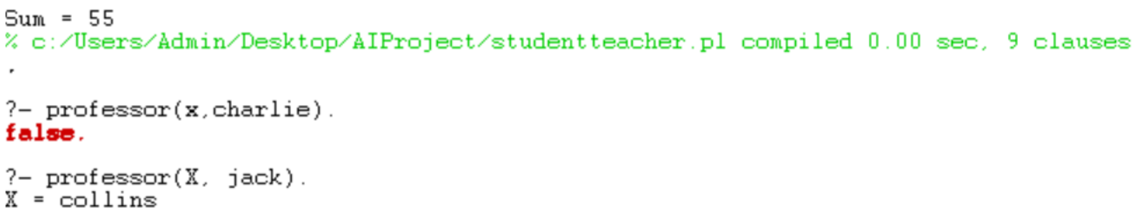
Day-4 Experiments

19.Write a Prolog Program for STUDENT-TEACHER-SUB-CODE.

Program:

```
% Facts for students and the subjects they study
studies(charlie, csc135).
studies(olivia, csc135).
studies(jack, csc131).
studies(arthur, csc134).

% Facts for teachers and the subjects they teach
teaches(kirke, csc135).
teaches(collins, csc131).
teaches(collins, csc171).
teaches(juniper, csc134).

% Rule to find the professor of a student for a given subject
professor(X, Y) :-
   teaches(X, C),
   studies(Y, C).
```

Sample output:



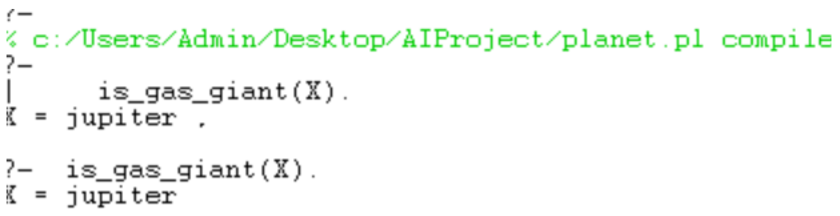
```
Sum = 55
% c:/Users/Admin/Desktop/AIProject/studentteacher.pl compiled 0.00 sec, 9 clauses
.

?- professor(x,charlie).
false.

?- professor(X, jack).
X = collins
```

20.Write a Prolog Program for PLANETS DB.

Program:

```
% Facts representing planets and their attributes
planet(mercury, terrestrial, 57.9, 0).
planet(venus, terrestrial, 108.2, 0).
planet(earth, terrestrial, 149.6, 1).
planet(mars, terrestrial, 227.9, 2).

planet(jupiter, gas_giant, 778.3, 79).

planet(saturn, gas_giant, 1427, 82).

planet(uranus, gas_giant, 2871, 27).

planet(neptune, gas_giant, 4497, 14).

planet(pluto, dwarf, 5913, 5).

% Rule to find if a planet is a gas giant

is_gas_giant(X) :-

   planet(X, gas_giant, _, _).

% Rule to find the planet with the most moons

planet_with_most_moons(Planet) :-

   planet(Planet, _, _, Moons),

   not((planet(_, _, _, Moons2), Moons2 > Moons)).

% Rule to find planets that are closer to each other (within a certain range)

planets_close_to_each_other(Planet1, Planet2, MaxDistance) :-

   planet(Planet1, _, Distance1, _),

   planet(Planet2, _, Distance2, _),

   Distance1 =\= Distance2,  % Ensure they are not the same planet

   abs(Distance1 - Distance2) =< MaxDistance.
```

Sample output:



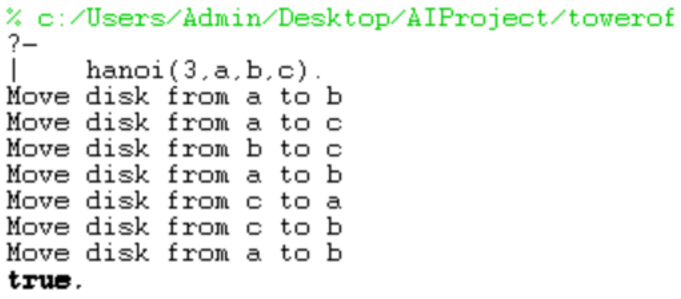
```
?-
% c:/Users/Admin/Desktop/AIProject/planet.pl compile
?-
|      is_gas_giant(X).
X = jupiter .

?-  is_gas_giant(X).
X = jupiter
```

21.Write a Prolog Program to implement Towers of Hanoi.

Program:

```
hanoi(0, _, _, _) :- !.

hanoi(N, Source, Target, Auxiliary) :-

   N > 0,
   M is N - 1,
   hanoi(M, Source, Auxiliary, Target),
   write('Move disk from '), write(Source), write(' to '), write(Target), nl,
   hanoi(M, Auxiliary, Target, Source).
```

Sample output:



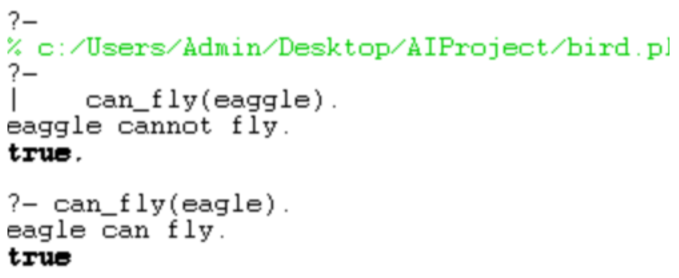
```
% c:/Users/Admin/Desktop/AIProject/towerof
?-
|    hanoi(3,a,b,c).
Move disk from a to b
Move disk from a to c
Move disk from b to c
Move disk from a to b
Move disk from c to a
Move disk from c to b
Move disk from a to b
true.
```

22.Write a Prolog Program to print particular bird can fly or not. Incorporate required queries.

Program:

```
bird(eagle).
bird(sparrow).
bird(penguin).
fly(penguin) :- !, fail.
fly(X) :- bird(X).
can_fly(Bird) :-
    fly(Bird),
    write(Bird), write(' can fly.'), nl.
can_fly(Bird) :-
    \+ fly(Bird),
    write(Bird), write(' cannot fly.'), nl.
```

Sample output:



```
?-
% c:/Users/Admin/Desktop/AIProject/bird.pl
?-
|    can_fly(eaggle).
eaggle cannot fly.
true.

?- can_fly(eagle).
eagle can fly.
true
```

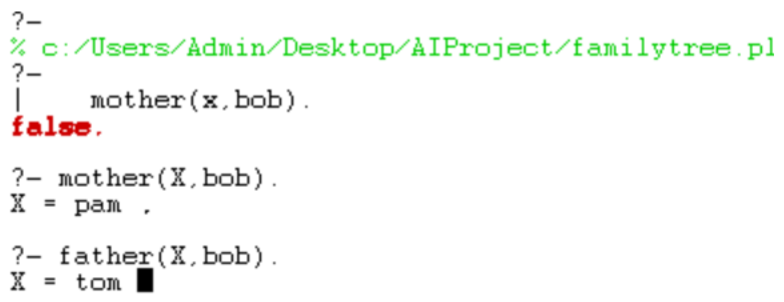
23.Write the Prolog program to implement family tree.

Program:

```
female(pam).
female(liz).
female(ann).
female(pat).
male(tom).
male(bob).
male(jim).
parent(pam, bob).
parent(tom, bob).
parent(tom, liz).
parent(bob, ann).
parent(bob, pat).
parent(liz, jim).
mother(X, Y) :-
   female(X),
   parent(X, Y).
father(X, Y) :-
   male(X),
   parent(X, Y).
grandfather(X, Y) :-
   male(X),
   parent(X, Z),
   parent(Z, Y).
```

```
grandmother(X, Y) :-
   female(X),
   parent(X, Z),
   parent(Z, Y).
sister(X, Y) :-
   female(X),
   parent(Z, X),
   parent(Z, Y),
   X \= Y.
brother(X, Y) :-
   male(X),
   parent(Z, X),
   parent(Z, Y),
   X \= Y.
```

Sample output:



```
?-
% c:/Users/Admin/Desktop/AIProject/familytree.pl
?-
|     mother(x,bob).
false.

?- mother(X,bob).
X = pam .

?- father(X,bob).
X = tom
```

24.Write a Prolog Program to suggest Dieting System based on Disease.

Program:

```
% Facts: Diseases and corresponding dieting recommendations
disease).
disease(hypertension).
disease(obesity).
disease(cancer).


% Dieting recommendations based on disease
```

```
diet(diabetes, 'Low-sugar, high-fiber diet').
diet(hypertension, 'Low-sodium, high-potassium diet').
diet(obesity, 'Low-calorie, high-protein diet').
diet(cancer, 'Balanced diet with emphasis on vitamins and minerals').

% Rule to suggest diet based on disease
suggest_diet(Disease) :-
   disease(Disease),
   diet(Disease, Diet),
   write('For ', Disease), write(', the recommended diet is: '), write(Diet), n
```