Name:- ATRIJ ROY

Section:A2

Roll No:- 002311001086

Assignment 9

—---

Domain 1

```
% Facts
parent(alice, bob).
parent(bob, carol).
parent(bob, carolina).
parent(eve, frank).

% Rules
grandparent(X, Z) :- parent(X, Y), parent(Y, Z).

ancestor(X, Z) :- parent(X, Z).
ancestor(X, Z) :- parent(X, Y), ancestor(Y, Z).
```

```
?- ['family_86.pl'].
true.

?- parent(eve,frank).
true.

?- parent(alice,bob).
true.

?- grandparent(alice,Who).
Who = carol ;
Who = carolina.

?- ancestor(alice,Who).
Who = bob ;
Who = carol ;
Who = carolina ;
false.
```

Domain 2

mammal(dog).
mammal(cat).
bird(sparrow).
bird(parrot).
fish(goldfish).

has_feathers(X) :- bird(X).
has_fur(X) :- mammal(X).
can_fly(X) :- bird(X), X \= penguin.

% test examples
is_animal(X) :- mammal(X) ; bird(X) ; fish(X).

```
?- ['animal_86.pl'].
true.

?- can_fly(X).
X = sparrow ;
X = parrot.

?- has_fur(X).
X = dog ,

?- has_fur(X).
X = dog ;
X = cat.

?- is_animal(X).
X = dog ;
X = cat ;
X = sparrow ;
X = parrot ;
X = goldfish.

?- is_bird(penguin).
ERROR: Unknown procedure:
^  Exception: (4) setup_ca
p
?- can_fly(penguin).
false.
```

Domain 3

author('Harry Potter', rowling).
author('The Hobbit', tolkien).
author('The Silmarillion', tolkien).
author('Pride and Prejudice', austen).

genre('Harry Potter', fantasy).

```
genre('The Hobbit', fantasy).
genre('Pride and Prejudice', romance).

same_author(Book1, Book2) :-
   author(Book1, A), author(Book2, A), Book1 \= Book2.

same_genre(Book1, Book2) :-
   genre(Book1, G), genre(Book2, G), Book1 \= Book2.
```

```
?- ['book_86.pl'].
true.

?- author('Harry Potter',Who)
|    .
Who = rowling.

?- same_author('The Hobbit',What)
|    .
What = 'The Silmarillion'.

?- same_genre('Pride and Prejudice',What).
false.

?- same_genre('Harry Potter',What).
What = 'The Hobbit'.

?-
```

Problem2

Code:-

```
% Employees
employee(harry).
employee(frank).
employee(eve).
employee(carol).
employee(dave).
employee(smith).

% Management relationships
manager(harry, frank).
manager(harry, carol).
manager(frank, dave).
manager(carol, eve).
manager(carol, smith).

% Project assignments
works_on(harry, alpha).
```

```prolog
works_on(frank, alpha).
works_on(eve, beta).
works_on(carol, beta).
works_on(dave, gamma).
works_on(smith, gamma).

% Rules

% Manager Details
% direct manager
is_manager_of(M, E) :- manager(M, E).

% indirect manager
is_manager_of(M, E) :-
   manager(M, X),
   is_manager_of(X, E).

% Team Members
team_members(M, Team) :-
   findall(E, is_manager_of(M, E), Team).

% Common Projects
common_projects(E1, E2, Projects) :-
   findall(P, (works_on(E1, P), works_on(E2, P)), Projects).

% Top Manager
top_manager(M) :-
   manager(M, _),       % M is a manager
   \+ manager(_, M).    % nobody manages M
```

Output:-

```
?- ['Problem2.pl'].
true.

?- team_members(harry,T).
T = [frank, carol, dave, eve, smith]

?- common_projects(harry,frank,P).
P = [alpha].

?- top_manager(M).
M = harry ;
M = harry ;
false.

?- common_projects(dave,smith,P).
P = [gamma].

?- is_manager_of(harry, dave).
true ;
false.

?- is_manager_of(eve,frank).
false.

?- is_manager_of(carol,smith).
true ;
false.
```

Explanation:-

This Prolog program models an organization’s employees, their management hierarchy, and project assignments. The knowledge base defines six employees — Harry, Frank, Eve, Carol, Dave, and Smith — along with their direct management relationships and project allocations. Facts such as `manager(harry, frank)` and `works_on(carol, beta)` represent the structure of the organization and the projects each employee is working on.

The program includes rules to derive useful information from this knowledge. The `is_manager_of/2` rule determines if a person is a direct or indirect manager of another, using recursion to handle indirect relationships. The `team_members/2` rule collects all employees under a manager into a list, while `common_projects/3` identifies projects shared by two employees. The `top_manager/1` rule finds the manager who has no superiors, using negation to ensure no one manages them.

Queries allow users to explore the knowledge base interactively. For example, `is_manager_of(harry, dave)` checks if Harry manages Dave, `team_members(harry, T)` lists all employees under Harry, `common_projects(frank, harry, P)` returns projects shared by Frank and Harry, and `top_manager(M)` identifies the organization’s highest-level manager. These queries demonstrate how the rules and facts can be combined to extract meaningful relationships in the organization.