CODE1: Use of simple facts

```
animal(elephant).
animal(monkey).
animal(cat).
animal(rat).
bigger(elephant, monkey).
bigger(monkey, cat).
bigger(cat, rat).
```

QUERIES and OUTPUTS:

```
| ?- animal(cat).
yes
| ?- bigger(cat, rat).
yes
| ?- bigger(monkey, rat).
no
| ?- bigger(rat, X).
no
| ?- bigger(cat, X).
X = rat
yes
| ?-
```

CODE2: Series and Parallel Resistance

```
series(R1,R2,Re) :- Re is R1 + R2.
parallel(R1,R2,Re) :- Re is ((R1 * R2) / (R1 + R2)).
```

QUERIES and OUTPUTS:

```
| ?- parallel(10,40,R3).
R3 = 8.0
yes
| ?- series(8,12,R4).
R4 = 20
yes
| ?- parallel(10,40,R3),series(R3,12,R4),parallel(R4,30,R5).
R3 = 8.0
R4 = 20.0
R5 = 12.0
yes
```

CODE3: Finding Min Max

```
% use of rules using if (:-) symbol.
% ! is a cut operator which skips other alternatives if fact in that line is true
```

```
find_max(X, Y, X) :- X >= Y, !.
find_max(X, Y, Y) :- X < Y.

find_min(X, Y, X) :- X =< Y, !.
find_min(X, Y, Y) :- X > Y.
```

QUERIES and OUTPUTS:

```
| ?- find_min(2, 4, X).
X = 2
yes
| ?- find_max(2, 5, X).
X = 5
yes
| ?-
```

CODE4: Linking Facts and Rules

```
%Facts and their English meanings
food(burger).    // burger is a food
food(sandwich).          // sandwich is a food
food(pizza).      // pizza is a food
lunch(sandwich).          // sandwich is a lunch
food(sandwich).          // sandwich is a food
dinner(pizza).    // pizza is a dinner
%Rule and meaning
meal(X) :- food(X). // Every food is a meal OR Anything is a meal if it is a food
```

QUERIES and OUTPUTS:

```
| ?- meal(sandwich).
yes
| ?- meal(pizza).
yes
| ?- meal(burger).
yes
| ?-
```

CODE5: Likes and Food Linking

```
/*
English statements:
1. mary like food
2. mary likes wine
3. mary likes john
4. joseph likes himself
write facts using above statements and also write rules for following
1 john likes anything that mary likes
2. john likes everyone who likes wine
```

```
3. john likes anyone who likes themselves
*/

likes(mary,food).
likes(mary,wine).
likes(mary,john).
likes(joseph,joseph).

likes(John, X) :- likes(mary, X).
likes(john, Someone) :- likes(Someone, wine).
likes(john, Someone) :- likes(Someone, Someone).
```

QUERIES and OUTPUTS:

```
| ?- likes(john, X).
X = food ? ;
X = wine ? ;
X = john ? ;
X = mary ? ;
X = joseph ?
Yes
| ?- likes(john, john).
true ?
yes
| ?-
```

CODE6: Student and Teacher

```
% Facts
studies(amol, 135).
studies(omkar, 135).
studies(javed, 131).
studies(ashwin, 134).

teaches(karthik, 135).
teaches(chahal, 131).
teaches(jasprit, 134).
teaches(chahel, 171).

% Rules and thier meaning
professor(X,Y) :- teaches(X, S), studies(Y, S).      % X teaches S and Y studies S
student(X,Y) :- professor(Y,X).                      % ":-" - means , "," - and
```

QUERIES and OUTPUTS:

```
Which subjects does amol study?
| ?- studies(amol, X)
X = 135
```

yes
Which subjects are taught by professor Chahal?
| ?- teaches(chahal, X).
X = 131
yes
Who are the students of Professor Karthik?
| ?- professor(Karthik, X).
X = amol
yes

CODE7: Family

```
% Facts
% By Gender
female(catherine).
female(sophia).
female(elizabeth).
male(X) :- \+ female(X).

% By Hierarchy
parent(james1, charles1).
parent(james1, elizabeth).
parent(charles1, catherine).
parent(charles1, james2).
parent(charles1, charles2).
parent(elizabeth, sophia).
parent(sophia, george).

% Rules
mother(X,Y) :- parent(X, Y), female(X).
father(X, Y) :- parent(X, Y), male(X).
sibling(X, Y) :- parent(P, X), parent(P, Y).
aunt(X, Y) :- female(X), parent(P, Y), sibling(X, P).
uncle(X, Y) :- male(X), parent(P, Y), sibling(X, P).
grandparent(X, Y) :- parent(P, Y), parent(X, P).
cousin(X, Y) :- parent(P, Y), sibling(S, P), parent(S, X).
```

QUERIES and OUTPUTS:

```
| ?- cousin(X, Y).
X = james2
Y = catherine ? ;
X = charles2
Y = catherine ? ;
X = sophia
Y = catherine ? ;
```