ARTIFICIAL INTELLIGENCE LAB EXPERIMENTs

**1.**

1. dog(fido).
2. dog(rover).
3. dog(henry).
4. cat(felix).
5. cat(jane).
6. animal(X):-dog(X).

**Answer queries :**

1. Find names of the dog.
2. Find names of the cat.
3. Who is the animal?

```
compiling C:/GNU-Prolog/bin/P01.pl for byte code...
C:/GNU-Prolog/bin/P01.pl:6: warning: discontiguous predicate dog/1 - clause ignored
C:/GNU-Prolog/bin/P01.pl compiled, 5 lines read - 513 bytes written, 10 ms
| ?-

dog(X).

X = fido ? a

X = rover

X = henry

yes

cat(X).

X = felix ? a

X = jane

yes

animal(X).

X = fido ? a

X = rover

X = henry

(16 ms) yes
| ?-
```

**2.**

1. dog(fido).
2. dog(rover).
3. dog(tom).
4. large(fido).
5. large(rover).
6. large(bill).
7. cat(mary).
8. cat(bill).
9. largeanimal(X):-dog(X), large(X).

**Answer querie :**

Who is the large animal ?

```
largeanimal(X).

X = fido ? a

X = rover

no
```

**3.**

1. parent(charlie,james).
2. parent(elizabeth,james).
3. parent(george,sophia).
4. parent(catherine,sophia).
5. parent(charlie,kith).

**Answer queries :**

1) Was George the parent of Sophia.
2) Who are James parents?
3) Who are the childrens of Charlies?
4) Who is a parent of whom?

```
compiling C:/GNU-Prolog/bin/AI-Exp08/P03.pl
C:/GNU-Prolog/bin/AI-Exp08/P03.pl compiled,
| ?-

parent(george, sophia).

yes

| ?- parent(X, james).

X = charlie ? a

X = elizabeth
```

```
| ?- parent(charlie, X).

X = james ? a

X = kith
```

```
parent(X, Y).

X = charlie
Y = james ? a

X = elizabeth
Y = james

X = george
Y = sophia

X = catherine
Y = sophia

X = charlie
Y = kith
```

**4.**

likes(john, susie).
likes(X, susie).
likes(john, X).
likes(X, john).
likes(john, susie).
likes(john, mary).
not(likes(john, pizza)).
likes(john, mary) :- likes(john, susie).

**Answer queries :**

i) Does John likes Susie?
ii) Whom does John likes?

```
| ?- likes(john, susie).

true ?

yes
| ?-
```

```
| ?- likes(john, X).

X = susie ? a

X = susie

true

X = john

X = susie

X = mary
```

**5.**

1. cat(tom).
2. loveseat(kunal, pasta).
3. hair(black).
4. lovesplay(nawaz, games).
5. lazy(pratyusha).
6. dances(lily).
7. closed(school).
8. free(ryan).
9. searching(tom, food).
10. happy(lily) :- dances(lily).
11. hungry(tom) :- searching(tom, food).
12. friends(jack, bili) :- lovesplay(jack, cricket) , lovesplay(bili, cricket).
13. go(ryan, play) :- closed(school), free(ryan).

**Answer queries :**

i) Who loves to eat pasta?
ii) Who is lazy?
iii) Who loves to play game?
iv) Is Lili happy?
v) Will Ryan go to play?

```
lovesplay(X, games).

X = nawaz

yes
| ?- happy(lili).

yes
```

```
| ?- loveseat(X, pasta).

X = kunal

yes
| ?-
lazy(X).

X = pratyusha

yes
go(ryan, play).

yes
| ?-
```

**6.**

1. parent(pam,bob).
2. parent(tom,bob).
3. parent(tom,liz).
4. parent(bob,ann).
5. parent(bob,pat).
6. parent(pat,jim).

**Answer queries :**

1) Who is a grandparent of Jim? (Hint: break down the query into 2 predicates)

2) Who are Tom's grandchildren?
3) Do Ann and Pat have a common parent?
4) Who is Pat's parent?
5) Who is Pat's grandparent?
6) Does Liz have a child?

```
parent(X, jim), parent(Y, X).

X = pat
Y = bob ?

yes
| ?- parent(tom, X), parent(X, Y).

X = bob
Y = ann ? a

X = bob
Y = pat

no
| ?- parent(X, ann), parent(X, pat).

X = bob ?

yes
| ?-
```

```
parent(X, pat).

X = bob ?

yes
| ?- parent(X, pat), parent(Y, X).

X = bob
Y = pam ? a

X = bob
Y = tom

no
| ?-
parent(liz, X).

no
| ?-
```

**7.**

```
woman(jia).
man(john).
healthy(john).
healthy(jia).
wealthy(john).
traveller(X) :- healthy(X) , wealthy(X).
cantravel(X) :- traveller(X).
```

**Answer queries :**

1) Who can travel?
2) Who is healthy and wealthy?

```
| ?- cantravel(X).

X = john ?

yes
| ?- healthy(X) , wealthy(X).

X = john ?

yes
| ?-
```

**8.**

```
food(burger).
food(sandwich).
food(pizza).
lunch(sandwich).
dinner(pizza).
meal(X) :- food(X).
```

Answer queries:

- Is pizza a food?
- Which food is meal and lunch?
- Is sandwich a dinner?

```
food(pizza).

yes
| ?- meal(X) , lunch(X).

X = sandwich ?

yes
| ?- dinner(sandwich).

no
| ?-
```

**9.**

```
1. studies(charlie, csc135).
2. studies(olivia, csc135).
3. studies(jack, csc131).
4. studies(arthur, csc134).
5. teaches(kirke, csc135).
6. teaches(collins, csc131).
7. teaches(collins, csc171).
8. teaches(juniper, csc134).
9. professor(X, Y) :- teaches(X, C) , studies(Y, C).
```

Answer queries:

- What does charlie study?
- Who are the students of professor kirke.

```
studies(charlie, X).

X = csc135

yes
| ?- teaches(kirke, X) , studies(Y, X).

X = csc135
Y = charlie ? a

X = csc135
Y = olivia
```

**10.**

```
suffering(charlie, fever).
suffering(charlie, runningnose).
suffering(charlie, headache).
suffering(micky, cough).
suffering(micky, headache).
suffering(micky, runningnose).
suffering(micky, sneezing).
suffering(branda, fever).
suffering(branda, rash).
hypothesis(X, flu) :- suffering(X, headache), suffering(X, bodyache), suffering(X, fever),
suffering(X, runningnose), suffering(X, cough), suffering(X, conjunctives).
hypothesis(X, measles) :- suffering(X, fever), suffering(X, cough), suffering(X, conjunctives),
suffering(X, runningnose), suffering(X, rash).
```

hypothesis(X, whoopingcough) :- suffering(X, cough), suffering(X, sneezing), suffering(X, runningnose).

```
suffering(X, fever).

X = charlie ? a

X = branda

(15 ms) no
| ?- suffering(branda, X).

X = fever ? a

X = rash
```

```
suffering(X, headache) , suffering(X, runningnose).

X = charlie ?

yes
| ?- suffering(charlie, X) , suffering(micky, X).

X = runningnose ? a

X = headache
```

**11.**

man(marcus).
loyal(X, Y).
hate(X, Y).
pompeian(marcus).
roman(X) :- loyal(X, carsar) ; hate(X, caesar).
pompeian(X) :- roman(X).
ruler(caesar).
tryassassinate(marcus, caesar).
tryassassinate(X, Y) :- not(loyal(X, Y)).
people(X) :- man(X).

```
tryassassinate(X, Y), ruler(Y).

X = marcus
Y = caesar ?

yes
```

```
loyal(marcus, caesar).

yes
| ?- hate(marcus, X).

ves
```

**12.**

```
symptom(moderatecough, flu).
symptom(chills, flu).
symptom(severebodyache, flu).
symptom(runningnose, flu).
symptom(chills, chickenpox).
symptom(highfever, chickenpox).
symptom(mildbodyache, cold).
symptom(runningnose, cold).
symptom(chills, cold).
```

```
symptom(runningnose, X).

X = flu ? a

X = cold

yes
| ?- symptom(X, cold).

X = mildbodyache ? a

X = runningnose

X = chills
```

```
| ?- symptom(chills, X).

X = flu ? a

X = chickenpox

X = cold

yes
| ?- symptom(mildbosyache, X) ; symptom(runningnose, X).

X = flu ? a

X = cold

yes
| ?- symptom(X, cold) , symptom(X, flu).

X = runningnose ? a

X = chills
```