



CS331

Programming Language Lab

ASSIGNMENT - 3 PROLOG

Naman Anand

200101070



Que 1

QUESTION 1:

Cut Example: Prolog program to insert an element into a list, if it is not present in the list before. And if the list has the element before we will simply cut it. For the membership checking also, if the item is at the head part, we should not check further, so cut it, otherwise check into the tail part.

```
list_member(X,[X|_]) :- !.  
list_member(X,[_|TAIL]) :- list_member(X,TAIL).
```

```
list_append(A,T,T) :- list_member(A,T),!.  
list_append(A,T,[A|T]).
```

Query-1: ?- list_append(a,[a,b,c,d,e], L).

Query-2: ?- list_append(k,[a,b,c,d,e], L).


For tracing of the code, please refer:

https://www.tutorialspoint.com/prolog/prolog_examples_of_cuts.htm

Note: This is one of the traces that are possible. Other ways are also possible.

Ans.

Code :

 que1.pl

```
File Edit Browse Compile Prolog Pce Help  
que1.pl  
list_member(X,[X|_]) :- !.  
list_member(X,[_|TAIL]) :- list_member(X,TAIL).  
list_append(A,T,T) :- list_member(A,T),!.  
list_append(A,T,[A|T]).
```

Query Execution :

Query1 : Query-1: ?- list_append(a,[a,b,c,d,e], L).

With Trace :

```
?- trace.  
true.  
[trace] ?- list_append(a,[a,b,c,d,e], L).  
Call: (10) list_append(a, [a, b, c, d, e], _2838) ? creep  
Call: (11) list_member(a, [a, b, c, d, e]) ? creep  
Exit: (11) list_member(a, [a, b, c, d, e]) ? creep  
Exit: (10) list_append(a, [a, b, c, d, e], [a, b, c, d, e]) ? creep  
L = [a, b, c, d, e].
```

Query2 : Query-2: ?- list_append(k,[a,b,c,d,e], L).

With Trace :

```
[trace] ?- list_append(k,[a,b,c,d,e], L).
Call: (10) list_append(k, [a, b, c, d, e], _8270) ? creep
Call: (11) list_member(k, [a, b, c, d, e]) ? creep
Call: (12) list_member(k, [b, c, d, e]) ? creep
Call: (13) list_member(k, [c, d, e]) ? creep
Call: (14) list_member(k, [d, e]) ? creep
Call: (15) list_member(k, [e]) ? creep
Call: (16) list_member(k, []) ? creep
Fail: (16) list_member(k, []) ? creep
Fail: (15) list_member(k, [e]) ? creep
Fail: (14) list_member(k, [d, e]) ? creep
Fail: (13) list_member(k, [c, d, e]) ? creep
Fail: (12) list_member(k, [b, c, d, e]) ? creep
Fail: (11) list_member(k, [a, b, c, d, e]) ? creep
Redo: (10) list_append(k, [a, b, c, d, e], _8270) ? creep
Exit: (10) list_append(k, [a, b, c, d, e], [k, a, b, c, d, e]) ? creep
L = [k, a, b, c, d, e].
```

Que 2

QUESTION 2:

Here are some simple clauses. Consider these as the knowledge base.

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

What do the following queries yield?

Explain with English translations of each line/query.

Query-1 ?- likes(mary,food).

Query-2 ?- likes(john,wine).

Query-3 ?- likes(john,food).

Ans

Knowledge base :

```
que2.pl
likes(mary, food) .
likes(mary, wine) .
likes(john, wine) .
likes(john, mary) .
```

These following lines means :

mary likes food.

mary likes wine.

john likes wine.

john likes mary.

Queries :

```
?- likes(mary, food) .
true.

?- likes(john, wine) .
true.

?- likes(john, food) .
false.
```

Query 1 :

It is asking if mary likes food. It is a Direct fact hence it is **true**.

Query 2 :

It is asking if john likes wine. It is a Direct fact hence it is **true**.

Query 1 :

It is asking if john likes food. But no fact tells us about this hence it is **false**.

Que 3

QUESTION 3:

Backtracking Example

`eats(lion,goat).`

`eats(lion,deer).`

`eats(tiger,lamb).`

`eats(tiger,deer).`

`common(X,Y,Z):-eats(X,Z),eats(Y,Z).`

Write a Query to find out what is the common food which both lions and tigers eat?

Ans

Code :

```
que3.pl
eats(lion,goat).
eats(lion,deer).
eats(tiger,lamb).
eats(tiger,deer).
common(X,Y,Z):-eats(X,Z),eats(Y,Z).
```

Query :

TO FIND OUT WHAT IS THE COMMON FOOD WHICH BOTH LIONS AND TIGERS EAT?

⇒ `common(tiger,lion,Z).`

or

⇒ `common(lion,tiger,Z).`

```
?- common(tiger,lion,Z).
Z = deer.
```

```
?- common(lion,tiger,Z).
Z = deer.
```

We can also see that it is true as tiger eats deer and lion also eats deer according to our knowledge base.

Que 4

QUESTION 4:

Basic prolog queries

male(homer).

male(bart).

male(abe).

male(luke).

female(marge).

female(lisa).

female(maggie).

female(mona).

female(jane).

parent(homer, bart).

parent(homer, lisa).

parent(homer, maggie).

parent(marge, bart).

parent(marge, lisa).

parent(marge, maggie).

parent(abe, homer).

parent(mona, homer).

parent(luke, mona).

parent(jane, abe).

mother(X, Y) :- parent(X, Y), female(X).

father(X, Y) :- parent(X, Y), male(X).

son(X, Y) :- parent(Y, X), male(X).

daughter(X, Y) :- parent(Y, X), female(X).

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

Find out answers to the following queries:

Query-1: mother(X,maggie).

Query-2: son(X,mona).

Query-3: grandparent(luke,Y).

Query-4: grandparent(jane, Y).

Ans

Knowledge Base :

que4.pl

```
male(homer).
male(bart).
male(abe).
male(luke).
female(marge).
female(lisa).
female(maggie).
female(mona).
female(jane).
parent(homer, bart).
parent(homer, lisa).
parent(homer, maggie).
parent(marge, bart).
parent(marge, lisa).
parent(marge, maggie).
parent(abe, homer).
parent(mona, homer).
parent(luke, mona).
parent(jane, abe).
mother(X, Y) :- parent(X, Y), female(X).
father(X, Y) :- parent(X, Y), male(X).
son(X, Y) :- parent(Y, X), male(X).
daughter(X, Y) :- parent(Y, X), female(X).
grandparent(X, Y) :- parent(X, Z), parent(Z, Y).
```

Queries :

```
?- mother(X, maggie).
X = marge.

?- son(X, mona).
X = homer.

?- grandparent(luke, Y).
Y = homer.

?- grandparent(jane, Y).
Y = homer.
```

Query 1 :

It checks who is mother of maggie. We know if X is mother of maggie then X is parent of maggie and X is female. So after checking for all possible values we got X = marge.

Query 2 :

It checks who is son of mona. We know if X is son of mona then mona is parent of X and X is male. So after checking for all possible values we got X =homer.

Query 3 :

It checks luke is grandparent of whom. So after checking for all possible values we got Y =homer.

Query 4 :

It checks jane is grandparent of whom. So after checking for all possible values we got Y = homer.

Thank You