LAB 6 | Artificial Intelligence

Aim : Write a prolog program to check whether a number is a member of a given list or not.

1. Write a prolog program to check whether a number is a member of a given list or not.

Code :

```
domains
        list = integer*

predicates
        member(integer,list)

clauses
        member(X,[X|_]).
        member(X,[_|T]):-member(X,T).
```

Output:

```
Goal : member(1, [ 1, 2, 3 ] )
Yes

Goal : member(0, [ 1, 2, 3 ] )
No
```

2. Write a prolog program to concatenate two lists giving a third list.

Code :

```
domains
        list = integer*

predicates
        append(list,list,list).

clauses
        append([],L,L).
        append([X|L1],L2,[X|L3]):-append(L1,L2,L3).
```

Output:

```
Goal : append( [ 1, 2 ], [ 3, 4 ] ,X )
X=[ 1, 2, 3, 4 ]
1 Solution

Goal : append( [ 1, 2 ], [ 3, 4 ], [ 1, 2, 3, 4 ] )
Yes
```

3. Write a prolog program to find the last element in a given list.

Code:

```
domains
        list=integer*

predicates
        last_element(list,integer)

clauses
        last_element([Z],X):-Z=X.
        last_element([_|T],X):-last_element(T,X).
```

Output:

```
Goal : last_element( [ 1, 2, 3, 4 ], X )
X=4
1 Solution

Goal : last_element( [ 1, 2, 3 ], 3 )
Yes
```

4. Write a prolog program to reverse a list.

Code:

```
domains
        list=integer*

predicates
        reverse(list,list,list).

clauses
        reverse([],InputList,InputList).
        reverse([H|T],List1,List2):-reverse(T,[H|List1],List2).
```

Output:

```
Goal : reverse( [ 1, 2, 3 ], [ ], Z)
Z=[ 3, 2, 1 ]
1 Solution

Goal : reverse( [ 1, 2, 3 ], [ ], [ 2, 3 ] )
No
```

5. Write a prolog program to find the nth element of a list.

Code:

```
domains
        list=integer*

predicates
        nth_element(list,integer)

clauses
        nth_element([H|_],1):-write(H),nl.
        nth_element([_|T],N):-NN=N-1,
                              nth_element(T,NN).
```

Output:

```
Goal : nth_element( [ 10, 20, 30, 40, 50 ], 3 )
30
Yes
```

6. Write a prolog program to split a list in two lists such that one list contains negative numbers and one contains positive numbers.

Code:

```
domains
        list=integer*

predicates
        split_list(list,list,list).

clauses
        split_list([],[],[]).
        split_list([X|L],[X|L1],L2):-X>=0,!,split_list(L,L1,L2).
        split_list([X|L],L1,[X|L2]):-split_list(L,L1,L2).
```

Output:

```
Goal : split_list( [ 1, -1, -2, 2 ], P, N )
P=[ 1, 2 ], N=[ -1, -2 ]

Goal : split_list( [ 1, -1, -2, 2 ], [ 1, -2 ], [ 2, -1 ] )
No
```