# CER4C3
# Abstraction and Paradigms of Programming

**Bachelor of Engineering**

**Computer Engineering**

**II Year IV Sem**

# List Membership

To check whether an object X is a member of the list L.
list_member(X,L).
X is an element in the list
L is a list of elements

The query :
?-list_member(X,L) will hold true if the element X occurs in the list L.
For example – list_member(b,[a,b,c]). – True
list_member(p,[a,b,c]). – False

# List Membership

The program for list membership relation can be based on the following facts –

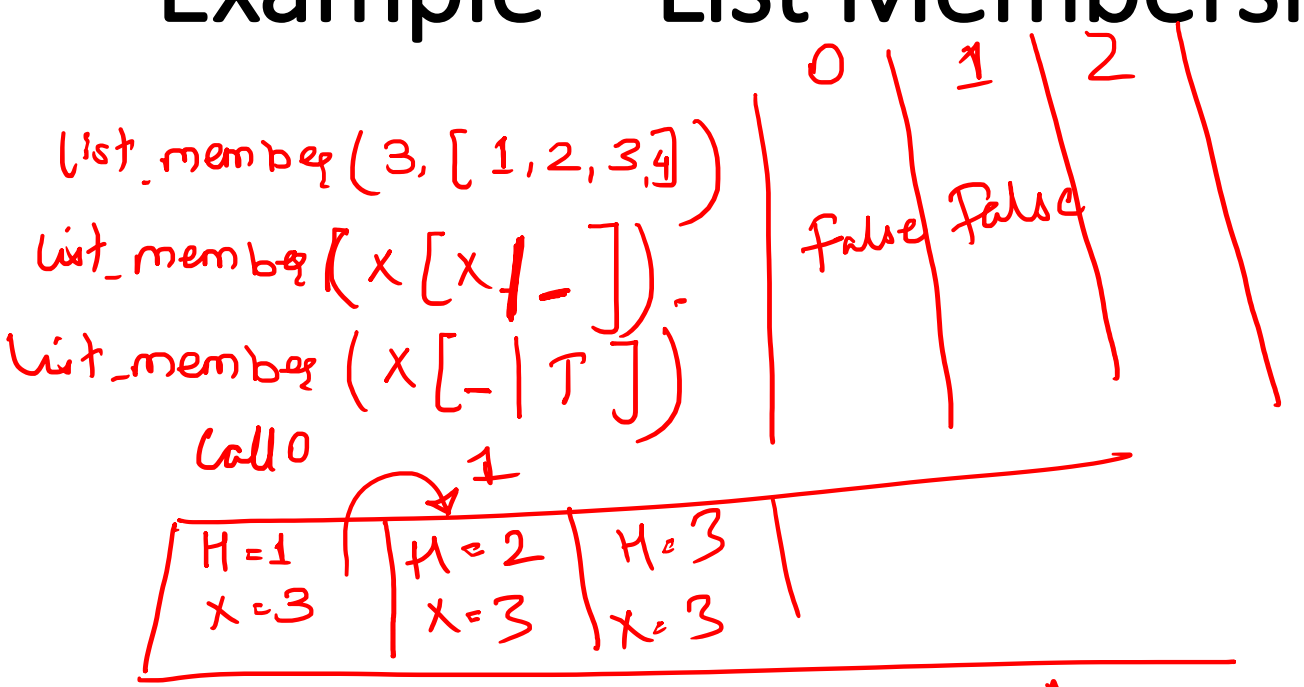
X is a member of the list L if –

Either element X is present in the head part of the list or,

The element X is present in the tail part of the list.

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

# Example – List Membership




Deciding the list in head, tail notation

[1 | [2,3,4]]

the tail is itself a list

∴ Recursion separate list

Recursion stack

Tail is itself a list

invoking the second statement,

invokes the entire procedure

Tail at 1 [2,3,4]

Tail at 2 [3,4]

True

# List Length Calculation

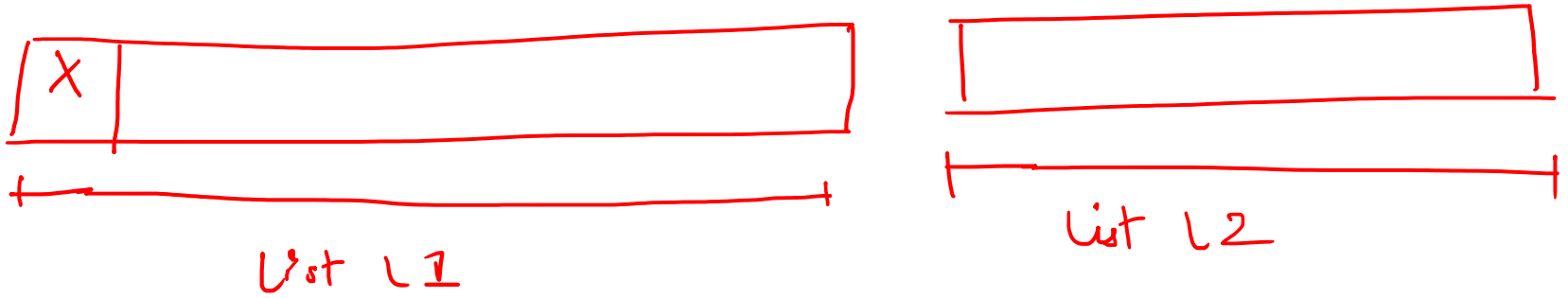
list_length(List,N).

It will be useful considering two cases –

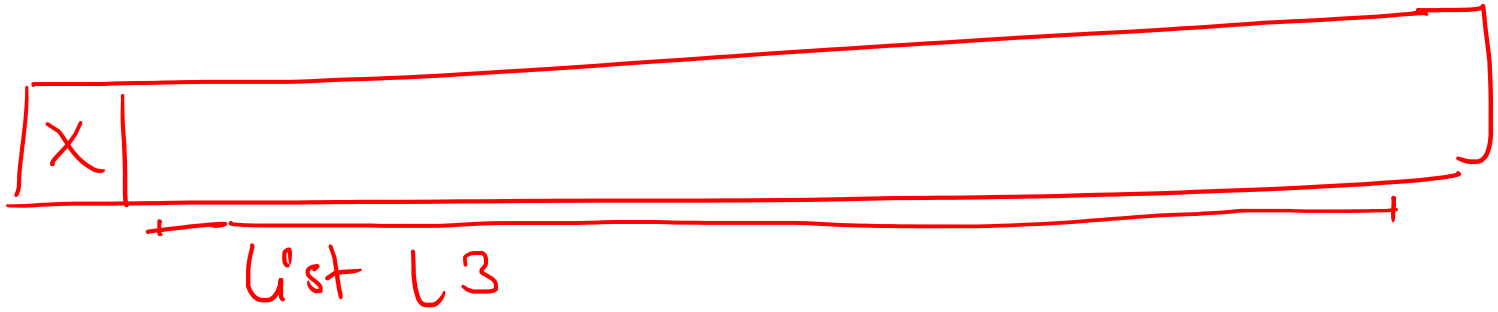
- If the list is empty then, the length of the list will be zero.
- If the list is represented in the head tail notation. As list[X|Y]
- Then the length of the list will be tail+1.

```
list_length([],0).
list_length([_|TAIL],N) :- list_length(TAIL,N1),
                           N is N1 +1.
```

# List Concatenation




Head = X




L3 = concatenation of L1 and L2.

# List Concatenation

Following can be the possible scenarios-

If we concatenate a blank list [] with a list L of N elements then, the concatenated will be the same list L with N elements.

concat([],L,L).

concat([X1|L1],L2,[X1|X3]) :-concat(L1,L2,L3).

# List Deletion

Deleting an item from the given list –

There can be following cases –

- Deleting an element from the list which is the only element then, after deletion we will be left with an empty list.
- If the item to be deleted is the head of the list then, the result after deletion will be the tail of the list.
- If X is in the tail then, it is deleted from there.

# List Deletion

```
del(Y,[Y],[]).
del(X,[X|List1],List1).
del(X,[Y|List1],[Y|List1]) :- del(X,List,List1).
```