Artificial Intelligence (CS308)

Assignment - 5

**U19CS012**

1.) Perform Towers of Hanoi.

**Prolog Code**

*% Base Case - Only 1 disk to be moved from A To C*

toh(1, A, C, *\_*) *:-*

    write(' Move Top Disk from '),

    write(A),

    write(' to '),

    write(C),

    nl*.*

*% Recursive toh defination to Move N disks from A to C using B {Helper}*

toh(N, A, C, B) *:-*

    N>1,

    M is N-1,

*% Move N-1 disks from src (A) to intermediate (B)*

    toh(M, A, B, C),

*% Move the Nth disk from src (A) to destination (C)*

    toh(1, A, C, *\_*),

*% Move the Remaining N-1 disks from intermediate (B) to destination (C)*

    toh(M, B, C, A)*.*

**Output**



2.) WAP to check whether the number is **present** in the list or not.

**Prolog Code**

*% If the Element is Equal to Head of the List*

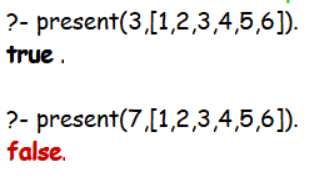
present(X, [X|*\_*])*.*

*% Recursive Call for Remaining List*

present(X, [*\_*|T]) *:-*

    present(X, T)*.*

**Output**

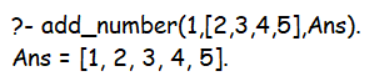


3.) WAP to add a number {in Front} of list.

**Prolog Code**

add\_number(X, L1, [X|L1])*.*

**Output**



4.) WAP to **Concat Two Lists** and store the result in **Third list**.

**Prolog Code**

*% Usage - append\_list(L1, L2, Concat\_List)*

*% If L1 is empty, resultant list will be equal to L2 (base case)*

append\_list([], L2, L2)*.*

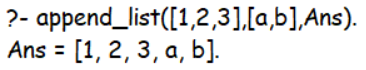
*% Recursive Defination of List Concatanation*

append\_list([H | T], L2, [H | L3]) *:-*

*% Tail & L2 will be Concatanated to L3*

    append\_list(T, L2, L3)*.*

**Output**



5.) WAP to delete an element from the list.

**Prolog Code**

*% Base Case - Remove X from List Containing [X], will lead to Empty List*

list\_del(X, [X], [])*.*

*% If X is Present in the Head, Delete it*

list\_del(X,[X|Tail], Tail)*.*

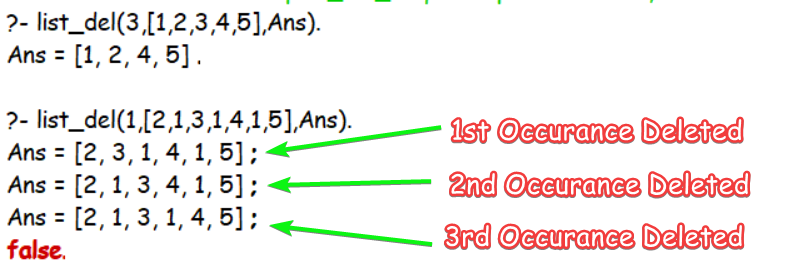
*% Recursive Function*

list\_del(X, [Head|Tail], [Head|NewTail]) *:-*

*% Recursively Braking List into Head & Tail*

    list\_del(X,Tail,NewTail)*.*

**Output**



6.) WAP to sum the elements of a list of numbers.

**Prolog Code**

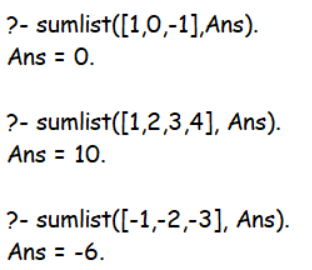
*% Base Case - Sum of Empty List = 0*

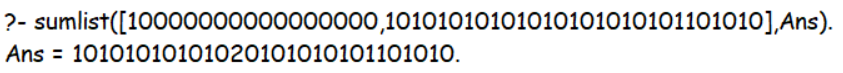
sumlist([],0)*.*

*% Recursive Defination*

sumlist([H|T],N) *:-* sumlist([T],N1), N is N1 + H*.*

**Output**





[Note: This also helps us to add **Really Big Numbers** as well!]

**SUBMITTED BY**: U19CS012

BHAGYA VINOD RANA