**Supplementary Material for Session 2**

1. **Recursion in Prolog:**
2. Ancestor
3. A parent is an ancestor.
4. A parent of an ancestor is an ancestor.

[X is an ancestor of Y, if X is a parent of an ancestor of Y.]

-------------------------

ancestor(X, Y) :- parent(X, Y), !.

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

1. Factorial of 0 is 1.

If n is greater than 0, then factorial of n is the product of n and the factorial of n-1.

[Factorial of N is F, if N is greater than 0 and F is the product of N and factorial of N-1.]

-----------------------

factorial(0, 1):-!.

factorial(N, F) :- N>0, N1 is N-1, factorial(N1, F1), F is N\*F1.

1. 100+105+110+ … +(100+(n-1)x5)

Sum of the 1st one element is 100.

Sum of the 1st n elements is the sum of 1st n-1 elements and the nth element, which is 100+(n-1)x5.

--------------------

sum1(1, 100):-!.

sum1(N, S):-N1 is N-1, sum1(N1, S1), S is S1+100+(n-1)x5.

1. Representing a weighted graph and finding the length of a path

neighbor(i,a,35). neighbor(i,b,45). neighbor(a,c,20).

neighbor(a,d,30). neighbor(b,d,25). neighbor(b,e,35).

neighbor(b,f,27). neighbor(c,d,30). neighbor(c,g,47).

neighbor(d,g,30). neighbor(e,g,25).

pathLength(X,Y,L):- neighbor(X,Y,L),!.

pathLength(X,Y,L):- neighbor(X,Z,L1), pathLength(Z,Y,L2), L is L1+L2.

1. **Working with Python:**

def ssum(N,I,F):

if (N==0):

return 1

elif (N>=1):

return ssum(N-1,S,F)+F+(N-1)\*I.

# Main

t=int(input("How many times?"))

for i in range(t):

print("Iteration:",i+1)

f=int(input("First element:"))

d=int(input("Interval:"))

n=int(input("n:"))

print("Series sum:%".format, ssum(n,d,f))

# Use of global variables in Python

def f():

global s

print (s)

s = "I love python!"

print (s)

# Global Scope

s = "Python is great!"

f()

print (s)

1. h3 in Prolog.

:-dynamic(hval/1).

/\* Evaluates a 8-queens' state given as list of 8 digits \*/

evalState(L,V):- assert(hval(0)),hl(1,L), di\_up(1,L),di\_dn(1,L),hval(V),

retractall(hval(\_)).

hl(8,\_):-!. hl(I,L):- nthel(I,L,X), chk\_incr(I,L,X), I1 is I+1, hl(I1,L).

chk\_incr(8,\_,\_):-!. chk\_incr(I,L,X):- I1 is I+1, nthel(I1,L,Y),

do\_incr(X,Y),chk\_incr(I1,L,X).

do\_incr(X,Y):- X=Y, incr\_hval. do\_incr(\_,\_).

incr\_hval:-hval(V), V1 is V+1, retract(hval(\_)), assert(hval(V1)).

di\_up(8,\_):-!. di\_up(I,L):- nthel(I,L,X), chkup\_incr(I,L,X,0), I1 is I+1,

di\_up(I1,L).

chkup\_incr(8,\_,\_,\_):-!.

chkup\_incr(I,L,X,K):- I1 is I+1, nthel(I1,L,Y), K1 is K+1,

doup\_incr(X,Y,K1), chkup\_incr(I1,L,X,K1).

doup\_incr(X,Y,K1):- X1 is X+K1, Y=X1, incr\_hval. doup\_incr(\_,\_,\_).

di\_dn(8,\_):-!. di\_dn(I,L):- nthel(I,L,X), chkdn\_incr(I,L,X,0), I1 is I+1,

di\_dn(I1,L).

chkdn\_incr(8,\_,\_,\_):-!.

chkdn\_incr(I,L,X,K):- I1 is I+1, nthel(I1,L,Y), K1 is K+1,

dodn\_incr(X,Y,K1), chkdn\_incr(I1,L,X,K1).

dodn\_incr(X,Y,K1):- X1 is X-K1, Y=X1, incr\_hval. dodn\_incr(\_,\_,\_).

% A procedure to find the nth element of a list

nthel(N,[\_|T],El):- N1 is N-1, nthel(N1,T,El).

nthel(1,[H|\_],H):-!.