

# ASSIGNMENT 1

NAME-HIMANSHU  
KUAMR

ROLL NO.-- RD2014A01

REGISTRATION

NUMBER— 12008223

COURSE CODE — CAP379:  
(ARTIFICIAL INTELLIGENCE  
LABORATORY)

(SET — A)

# QUE1. WRITE A PROLOG PROGRAM TO FIND THE FACTORIAL OF A NUMBER WITH THE HELP OF INPUT OUTPUT FUNCTIONS. IMPLEMENT THE PROGRAM IN PROLOG AND ATTACH OUTPUT SCREENSHOT ALSO

- Prolog Factorial function definition is also similar to a normal factorial function.
- Factorial(0,1) i.e., factorial of 0 is generally 1.
- Factorial(N,M), if any temporary value N1 is assigned to N-1.
- Factorial(N1,M1), and is factorial of N1 is F1.
- M is NM1 i.e., assigning M to N\*M1, then value of N is F.
- The above happens to be the recursive relation between N and factorial F. It reviews rules for particular relation in the top to bottom order.


```
loop.pl - Notepad
File Edit Format View Help

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

Factorial(4,24)

4>0	3 is 4-1	factorial(3,3)	24 is 4*6
3>0	2 is 3-1	factorial(2,2)	6 is 3*2
2>0	1 is 2-1	factorial(1,1)	2 is 2*1
1>0	0 is 1-1	factorial(0,1)	1 is 1*1

# OUTPUT OF THE CODE

 SWI-Prolog (AMD64, Multi-threaded, version 8.4.1)  
File Edit Settings Run Debug Help  
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.1)  
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.  
Please run ?- license. for legal details.  
  
For online help and background, visit <https://www.swi-prolog.org>  
For built-in help, use ?- help(Topic). or ?- apropos(Word).  
  
?-  
% c:/Users/himan/Desktop/AI pracital/loop.pl compiled 0.00 sec, 2 clauses  
?- factorial(4,24).  
**true** .  
  
?- factorial(4,Factorial).  
Factorial = 24  
Unknown action: + (h for help)  
Action? █

**QUE2. HOW MONKEY BANANA PROBLEM CAN BE IMPLEMENTED IN PROLOG? WRITE A PROLOG PROGRAM TO IMPLEMENT MONKEY BANANA PROBLEM. IMPLEMENT THE PROGRAM IN PROLOG AND ATTACH OUTPUT SCREENSHOT ALSO**

Initial state:

- monkey on ground
- with empty hand
- bananas suspended

Goal state:

- monkey eating

```
monkey.pl - Notepad
File Edit Format View Help
move(state(center,ontable,center,hasnot),
    grasp,
    state(center,ontable,center,has)).
move(state(P,onfloor,P,H),
    climb,
    state(P,ontable,P,H)).
move(state(P1,onfloor,P1,H),
    drag(P1,P2),
    state(P2,onfloor,P2,H)).
move(state(P1,onfloor,B,H),
    walk(P1,P2),
    state(P2,onfloor,B,H)).
canget(state(_,_,_,has)).
canget(State1) :-
    move(State1,_,State2),
    canget(State2).|
```



# SET OF OPERATION

- When the table is at the center, and monkey is on top of the table, and monkey does not have the banana (*ih***as not** state), then using the **grasp** action, it will change from **has not** state to **have** state.
- From the floor, it can move to the top of the table ( **on top** state), by performing the action **climb**.
- The **push** operation moves the table from one place to another.
- Monkey can move from one place to another using **walk** or **move** clauses.
- Another predicate will be canget(). Here we pass a state, so this will perform move predicate from one state to another using different actions, then perform canget() on state 2. When we have reached to the state '**has>**', this indicates '**has banana**'. We will stop the execution.

## OUTPUT OF THE PROGRAM

```
SWI-Prolog (AMD64, Multi-threaded, version 8.4.1)
File Edit Settings Run Debug Help
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.1)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?-
% c:/Users/himan/Desktop/AI pracital/monkey.pl compiled 0.00 sec, 6 clauses
?- canget(state(atdoor, onfloor, atwindow, hasnot)).
true ; [trace]
  Redo: (14) canget(state(center, ontable, center, has)) ? creep
  Call: (15) move(state(center, ontable, center, has), _6536, _6476) ? creep
  Fail: (15) move(state(center, ontable, center, has), _7294, _6476) ? creep
  Fail: (14) canget(state(center, ontable, center, has)) ? creep
  Redo: (13) move(state(_4274, onfloor, _4274, hasnot), _8802, _4286) ? creep
  Exit: (13) move(state(_4274, onfloor, _4274, hasnot), drag(_4274, _9506), state(_9506, onfloor, _9506, hasnot)) ? creep
  Call: (13) canget(state(_9506, onfloor, _9506, hasnot)) ? creep
  Call: (14) move(state(_9506, onfloor, _9506, hasnot), _11086, _11026) ? creep
  Exit: (14) move(state(_9506, onfloor, _9506, hasnot), climb, state(_9506, ontable, _9506, hasnot)) ? creep
  Call: (14) canget(state(_9506, ontable, _9506, hasnot)) ? creep
  Exit: (15) move(state(_9506, ontable, _9506, hasnot), _13364, _13304) ? creep
  Exit: (15) move(state(center, ontable, center, hasnot), grasp, state(center, ontable, center, has)) ? creep
  Call: (15) canget(state(center, ontable, center, has)) ? creep
  Exit: (15) canget(state(center, ontable, center, has)) ? creep
  Exit: (14) canget(state(center, ontable, center, hasnot)) ? creep
  Exit: (13) canget(state(center, onfloor, center, hasnot)) ? creep
  Exit: (12) canget(state(_4274, onfloor, _4274, hasnot)) ? creep
  Exit: (11) canget(state(atwindow, onfloor, atwindow, hasnot)) ? creep
true .

[trace] ?-
|
```