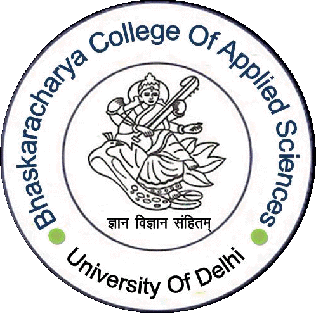
**Bhaskaracharya College of Applied Sciences**

**Artificial Intelligence**



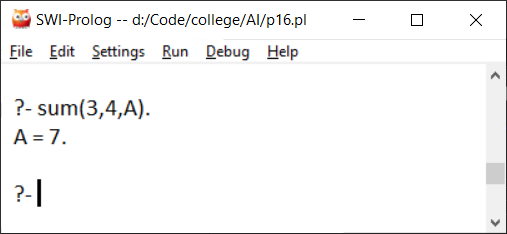
**Tushar Gautam**

**1602016**

**B.Sc.(H) Computer Science**

**%Q1:-Write a prolog program to calculate the sum of two numbers.**

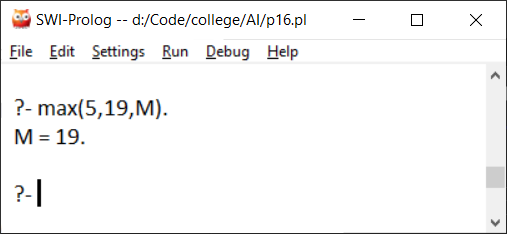
sum(A,B,C):- C is +(A,B).



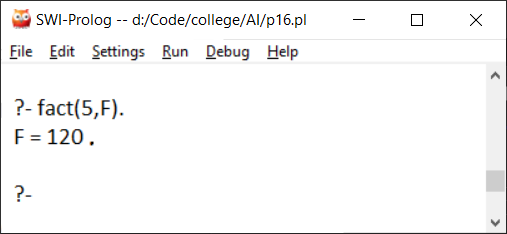
**%Q2:-Write a Prolog program to implement max(X, Y, M) so that M is the maximum of two numbers** **X and Y.**

max(X,Y,M):- X\=y,

X>Y -> M is X; M is Y.



**%Q3:- Write a program in PROLOG to implement factorial (N, F) where F represents the factorial of a number N.**

fact(0,1).

fact(N,F):- N>0,

N1 is N-1,

fact(N1,F1),

F is N\*F1.

**%Q4:- Write a program in PROLOG to implement generate\_fib(N,T) where T represents the Nth term of the Fibonacci series.**

generate\_fib(0,1).

generate\_fib(1,1).

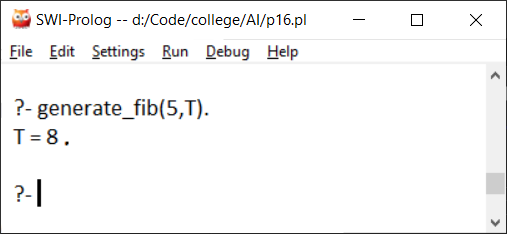
generate\_fib(N,T):-N1 is N-1,

generate\_fib(N1,T1),

N2 is N-2,

generate\_fib(N2,T2),

T is T1+T2.



**%Q5:- Write a Prolog program to implement GCD of two numbers.**

gcd(A,0,G):- G is A.

gcd(0,B,G):- G is B.

gcd(A,A,G):- G is A.

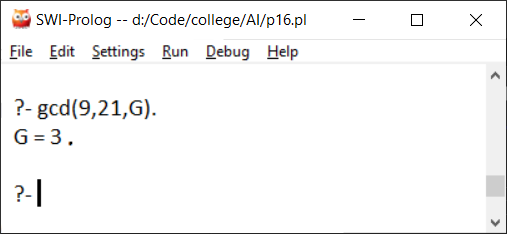
gcd(A,B,G):- B>A,

gcd(B,A,G).

gcd(A,B,G):- A>B,

A1 is A-B,

gcd(B,A1,G).



**%Q6:- Write a Prolog program to implement power (Num,Pow, Ans) : where Num is raised to the power Pow to get Ans.**

power(0,\_,not\_defined).

power(\_,0,Ans):- Ans is 1.

power(Num,Pow,Ans):-Pow<0,

Pow1 is \*(-1,Pow),

power(Num,Pow1,Ans1),

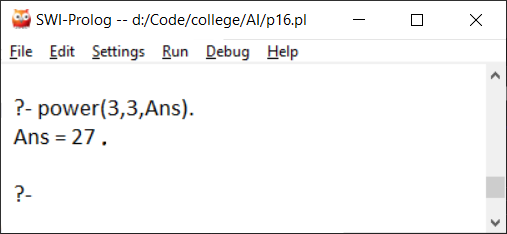
Ans is /(1,Ans1).

power(Num,Pow,Ans):- Pow>0,

Pow1 is Pow -1,

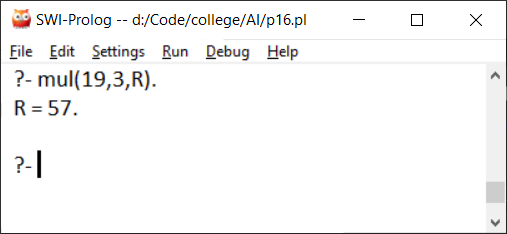
power(Num,Pow1,Ans1),

Ans is Ans1\*Num.



**%Q7:- Prolog program to implement multi (N1, N2, R) : where N1 and N2 denotes the numbers to be multiplied and R represents the result.**

mul(A,B,C):- C is \*(A,B).



**%Q8 :- Write a program in PROLOG to implement towerofhanoi (N) where N represents the number of discs.**

towerofhanoi(N):- power(2,N,A1),

A2 is -(A1,1),

write(N),write(" disks will require "),

write(A2),write(" steps to move from disk a to c with auxiliary b.\n"),

move(N,a,c,b).

move(1,X,Y,\_):-write('Move disk from '),

write(X),write(' to '),

write(Y),

nl.

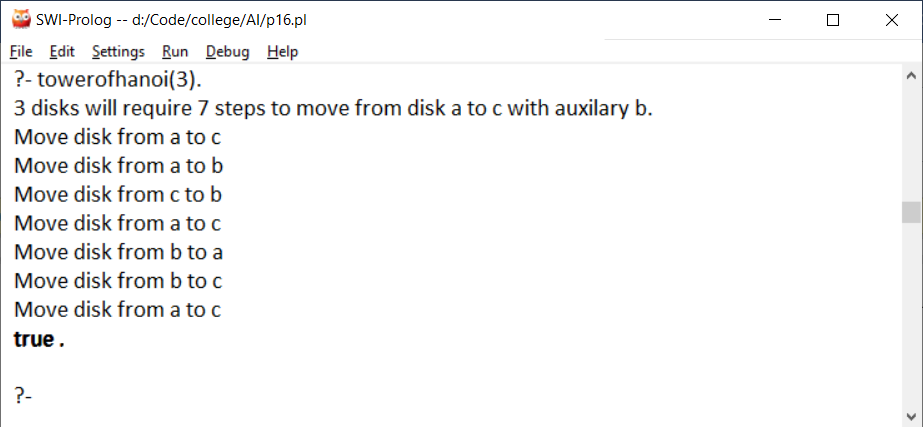
move(N,X,Y,Z):-N>1,

M is N-1,

move(M,X,Z,Y),

move(1,X,Y,\_),

move(M,Z,Y,X).



**/\* Q9:- Consider a cyclic directed graph [edge (p, q), edge (q, r), edge (q, r), edge (q, s), edge (s,t)] where edge (A,B) is a predicate indicating directed edge in a graph from a node A to a node B. Write a program to check whether there is a route from one node to another node.\*/**

edge(p,q).

edge(q,r).

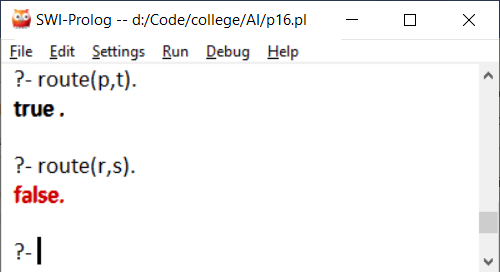
edge(q,s).

edge(s,t).

route(A,B):-edge(A,B).

route(A,B):- edge(A,C),

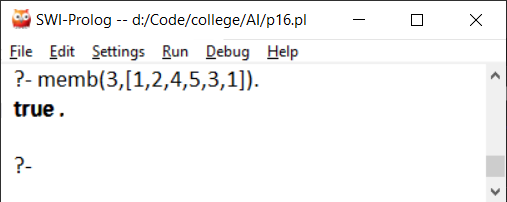
route(C,B).



**%Q10 :- Write a Prolog program to implement memb(X, L): to check whether X is a member of L or not.**

memb(X,[X|\_]).

memb(X,[\_|A]):-memb(X,A).

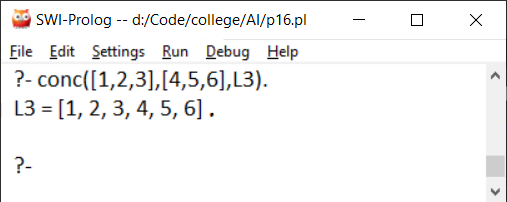


**%Q11:- Write a Prolog program to implement conc (L1, L2, L3) where L2 is the list to be appended with L1 to get the resulted list L3.**

conc([],[],[]).

conc([],[H2|T2],[H2|R]):-conc([],T2,R).

conc([H1|T1],L2,[H1|R]):-conc(T1,L2,R).

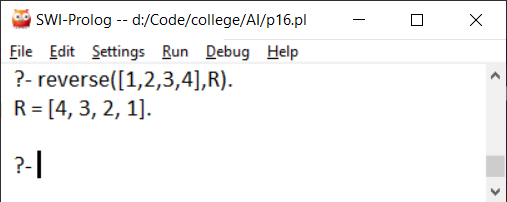


**%Q12:- Write a Prolog program to implement reverse (L, R) where List L is original and List R is reversed list.**

reverse([],[]).

reverse([A|B],R):-reverse(B,R1),

append(R1,[A],R).

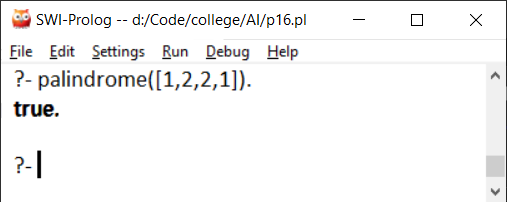


**%Q13:-Write a program in PROLOG to implement palindrome (L) which checks whether a list L is a palindrome or not.**

palindrome([]).

palindrome(A):-reverse(A,B),

A=B.



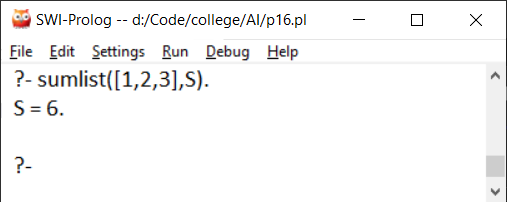
**%Q14:- Write a Prolog program to implement sumlist(L, S) so that S is the sum of a given list L.**

sumlist([],S):- S is 0.

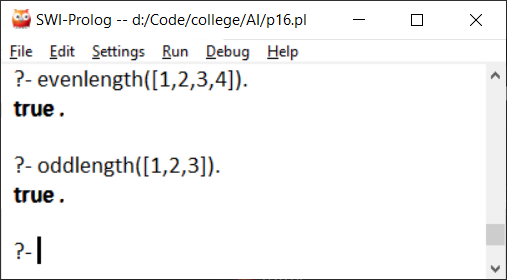
sumlist([X|Y],S):- X1 is X,

sumlist(Y,S1),

S is S1+X1.



**%Q15:- Write a Prolog program to implement two predicates evenlength(List) and oddlength(List) so that they are true if their argument is a list of even or odd length respectively.**

len([],0).

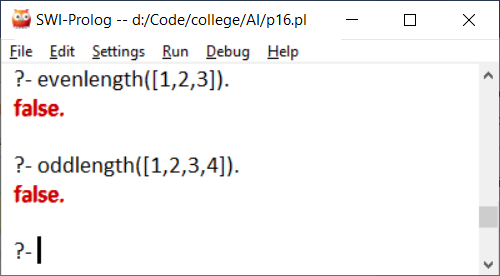
len([\_],B):- B is 1.

len([\_|Y],B):-len(Y,B1),

B is B1+1.

evenlength(L1):- len(L1,A),

gcd(A,2,B),

B=2.

oddlength(L1):- len(L1,A),

gcd(A,2,B),

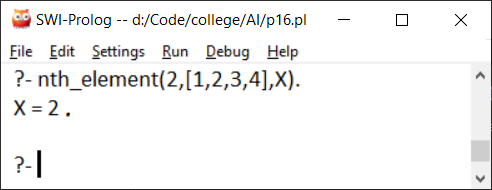
B=1.

**%Q16:-Write a Prolog program to implement nth\_element (N, L, X) where N is the desired position, L is a list and X represents the Nth element of L.**

nth\_element(1,[A|\_],X):- X is A.

nth\_element(N,[\_|B],X):- N1 is N-1,

nth\_element(N1,B,X).



**% Q17:-Write a program in PROLOG to implement remove\_dup (L, R) where L denotes the list with some duplicates and the list R denotes the list with duplicates removed.**

remove\_dup(L,R):- reverse(L,A),

remove\_dup1(A,B),

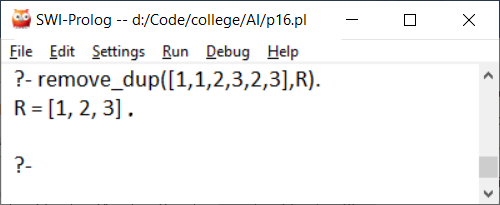
reverse(B,R).

remove\_dup1([],[]).

remove\_dup1([A|B],R):- memb(A,B),

remove\_dup1(B,R).

remove\_dup1([A|B],[A|C]):- remove\_dup1(B,C).

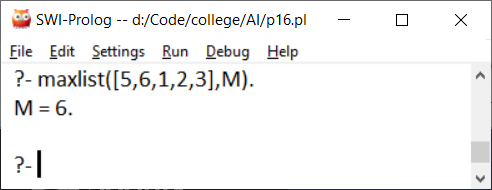


**%Q18:- Write a Prolog program to implement maxlist(L, M) so that M is the maximum number in the list.**

maxlist([L],L).

maxlist([X|Y],M):- maxlist(Y,M1),

X<M1 -> M is M1 ; M is X.

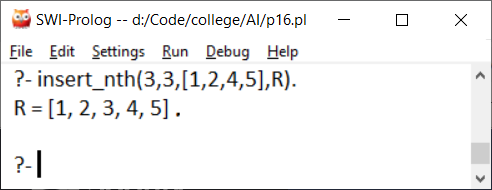


**%Q19:- Write a prolog program to implement insert\_nth(I, N, L, R) that inserts an item I into Nth position of list L to generate a list R.**

insert\_nth(I,1,L,[I|L]).

insert\_nth(I,N,[H|T],[H|R]):- N1 is N-1,

insert\_nth(I,N1,T,R).

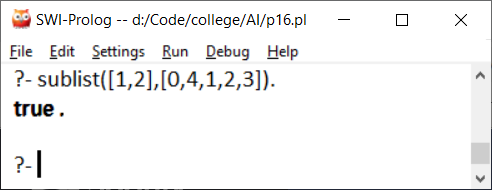


**%Q20 :- Write a Program in PROLOG to implement sublist(S, L) that checks whether the list S is the sublist of list L or not. (Check for sequence or the part in the same order).**

sublist([],\_).

sublist([H|T],[H|R]):- sublist(T,R).

sublist([H|T],[Q|R]):-sublist([H|T],R).

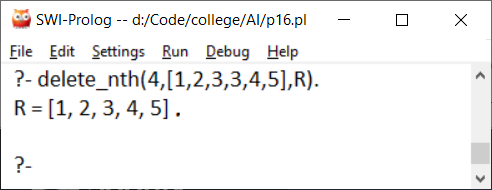


**%Q21:- Write a Prolog program to implement delete\_nth (N, L, R) that removes the element on Nth position from a list L to generate a list R.**

delete\_nth(1,[\_|T],T).

delete\_nth(N,[H|T],[H|R]):- N1 is N-1,

delete\_nth(N1,T,R).

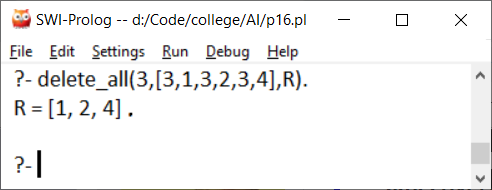


**%Q22 :- Write a program in PROLOG to implement delete\_all (X, L, R) where X denotes the element whose all occurrences has to be deleted from list L to obtain list R.**

delete\_all(\_,[],[]).

delete\_all(X,[X|T],R):-delete\_all(X,T,R).

delete\_all(X,[H|T],[H|R]):-delete\_all(X,T,R).



**%23 :- Write a program in PROLOG to implement merge (L1, L2, L3) where L1 is first ordered list and L2 is second ordered list and L3 represents the merged list.**

merge([],L2,L2).

merge(L1,[],L1).

merge([H1|T1],[H2|T2],[H1|R]):- H1=<H2,

merge(T1,[H2|T2],R).

merge([H1|T1],[H2|T2],[H2|R]):- H2<H1,

merge([H1|T1],T2,R).

