INDRAPRASTHA COLLEGE FOR WOMEN

UNIVERSITY OF DELHI

COURSE: BSC. (HONS.) COMPUTER SCIENCE

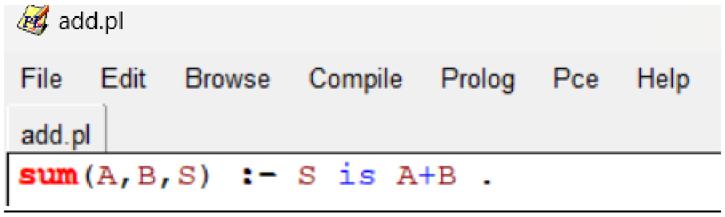
PRACTICAL: ARTIFICIAL INTELLIGENCE

SEMESTER:VI

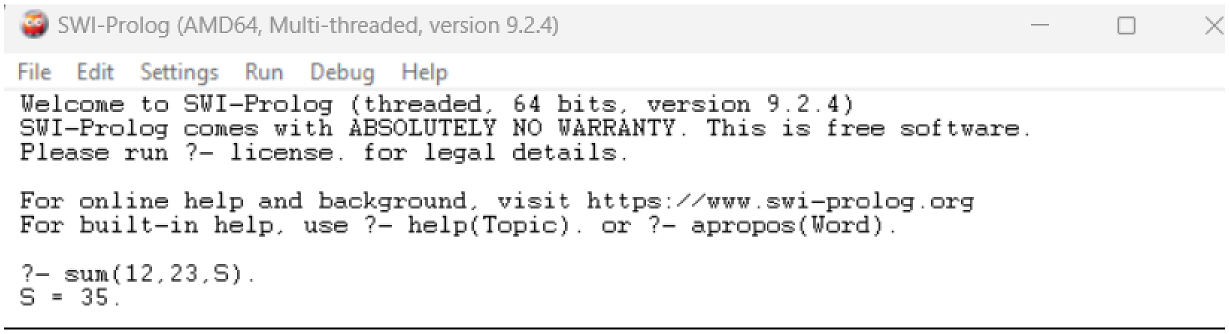
SUBMITTED TO: MR. SHAILENDRA SIR

SUBMITTED BY: TASHI LAMO
ROLL NO.:21/CS/54

Program 1: Write a prolog program to calculate the sum of two numbers.

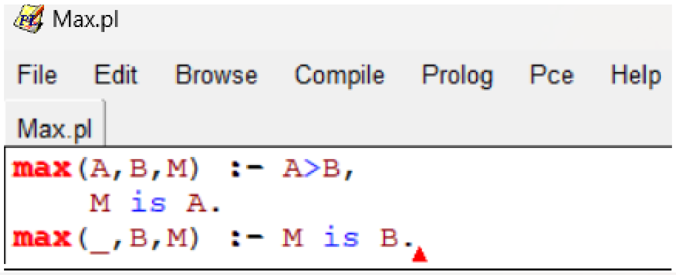


```
sum(A,B,S) :- S is A+B .
```

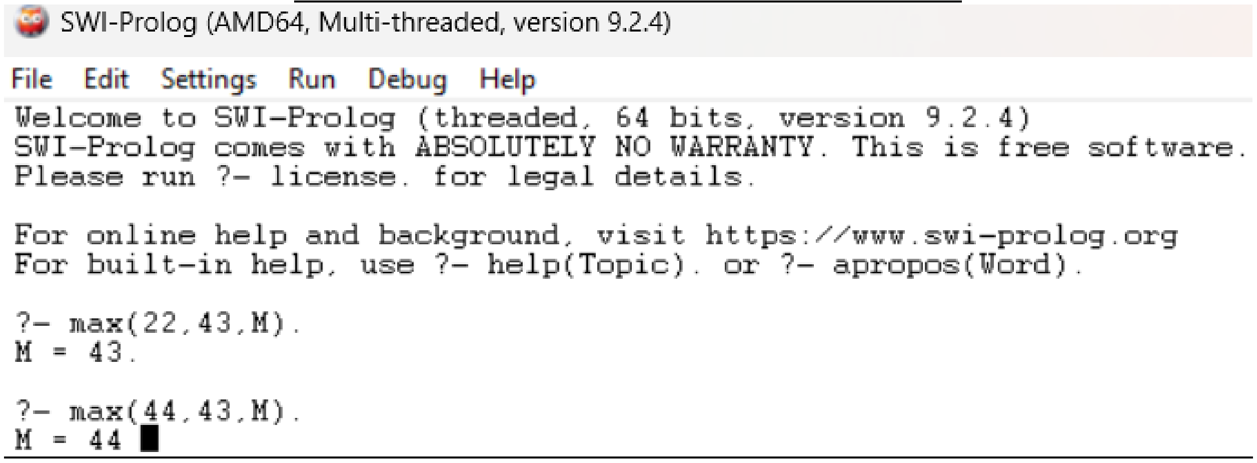

SWI-Prolog (AMD64, Multi-threaded, version 9.2.4)
File Edit Settings Run Debug Help

```
Welcome to SWI-Prolog (threaded, 64 bits, version 9.2.4)
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).

?- sum(12,23,S).
S = 35.
```

Program 2: Write a Prolog program to implement max(X, Y, M) so that M is the maximum of two numbers X and Y.

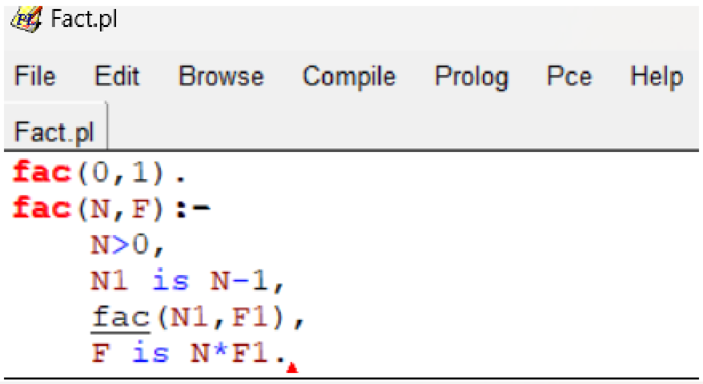


```
max(A,B,M) :- A>B,
    M is A.
max(_,B,M) :- M is B.
```

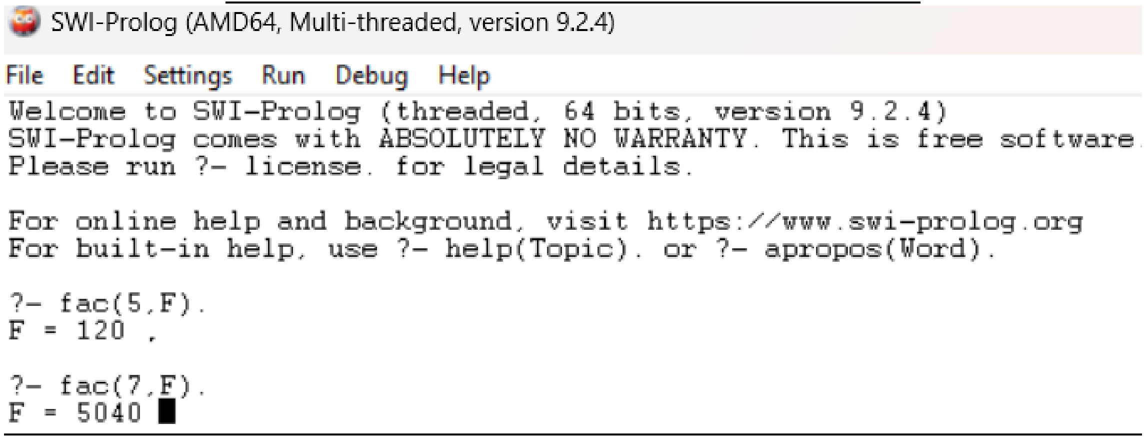

SWI-Prolog (AMD64, Multi-threaded, version 9.2.4)
File Edit Settings Run Debug Help

```
Welcome to SWI-Prolog (threaded, 64 bits, version 9.2.4)
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).

?- max(22,43,M).
M = 43.

?- max(44,43,M).
M = 44
```

Program 3: Write a program in PROLOG to implement factorial (N, F) where F represents the factorial of a number N.

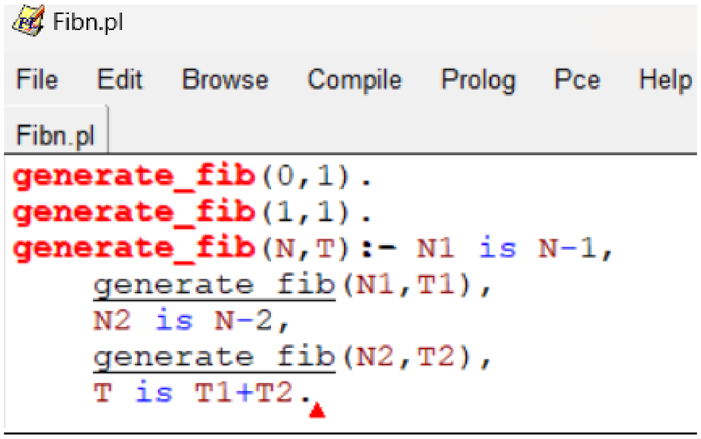


```
fac(0,1).
fac(N,F):-
    N>0,
    N1 is N-1,
    fac(N1,F1),
    F is N*F1.
```

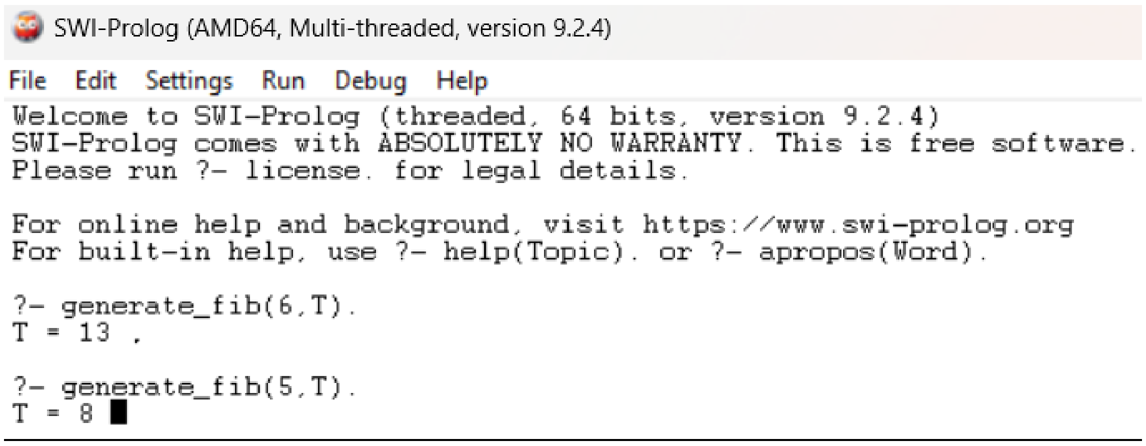

SWI-Prolog (AMD64, Multi-threaded, version 9.2.4)
File Edit Settings Run Debug Help

```
Welcome to SWI-Prolog (threaded, 64 bits, version 9.2.4)
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).

?- fac(5,F).
F = 120 ,

?- fac(7,F).
F = 5040
```

Program 4: Write a program in PROLOG to implement generate_fib(N,T) where T represents the Nth term of the fibonacci series.

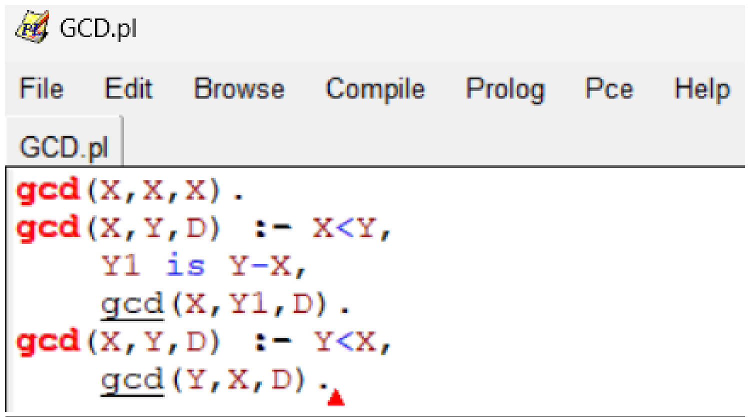


```
generate_fib(0,1).
generate_fib(1,1).
generate_fib(N,T):- N1 is N-1,
    generate_fib(N1,T1),
    N2 is N-2,
    generate_fib(N2,T2),
    T is T1+T2.
```

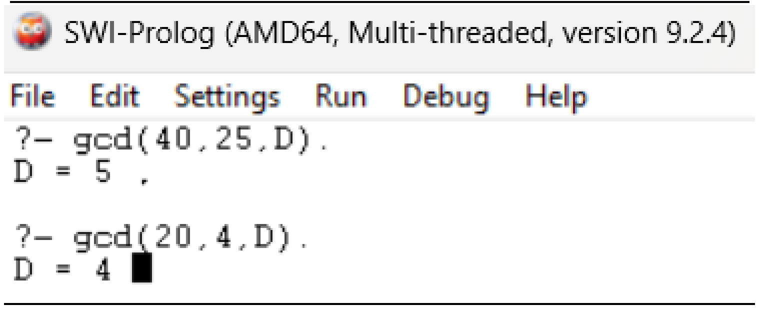

SWI-Prolog (AMD64, Multi-threaded, version 9.2.4)
File Edit Settings Run Debug Help

```
Welcome to SWI-Prolog (threaded, 64 bits, version 9.2.4)
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).

?- generate_fib(6,T).
T = 13 .

?- generate_fib(5,T).
T = 8
```

<u>Program 5: Write a Prolog program to implement GCD of two numbers.</u>

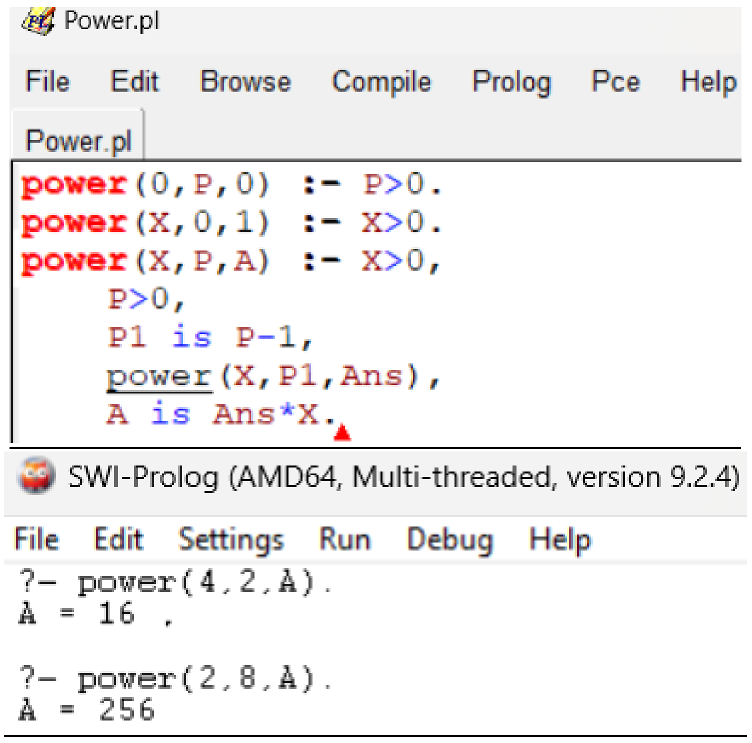


```
gcd(X,X,X).
gcd(X,Y,D) :- X<Y,
    Y1 is Y-X,
    gcd(X,Y1,D).
gcd(X,Y,D) :- Y<X,
    gcd(Y,X,D).
```


SWI-Prolog (AMD64, Multi-threaded, version 9.2.4)
File Edit Settings Run Debug Help

```
?- gcd(40,25,D).
D = 5 .

?- gcd(20,4,D).
D = 4
```

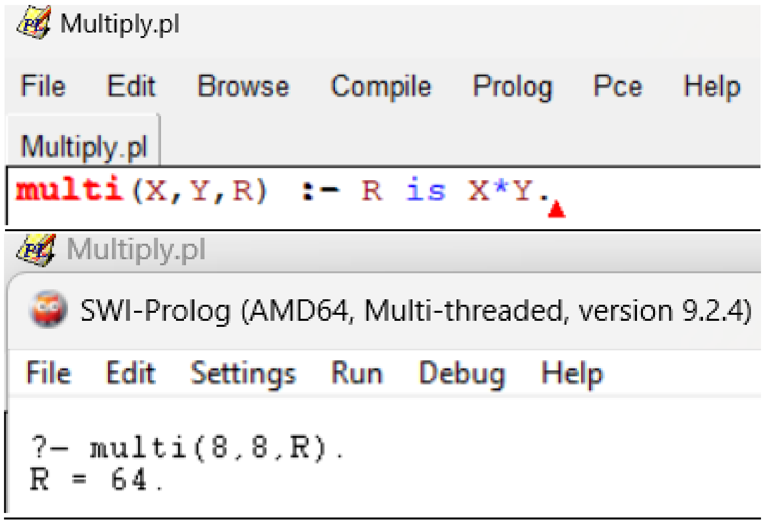
<u>Program 6: Write a Prolog program to implement power (Num,Pow, Ans) : where Num is raised to the power Pow to get Ans.</u>


Power.pl
File Edit Browse Compile Prolog Pce Help
Power.pl
SWI-Prolog (AMD64, Multi-threaded, version 9.2.4)
File Edit Settings Run Debug Help

```
power(0,P,0) :- P>0.
power(X,0,1) :- X>0.
power(X,P,A) :- X>0,
    P>0,
    P1 is P-1,
    power(X,P1,Ans),
    A is Ans*X.
```

```
?- power(4,2,A).
A = 16 .

?- power(2,8,A).
A = 256
```

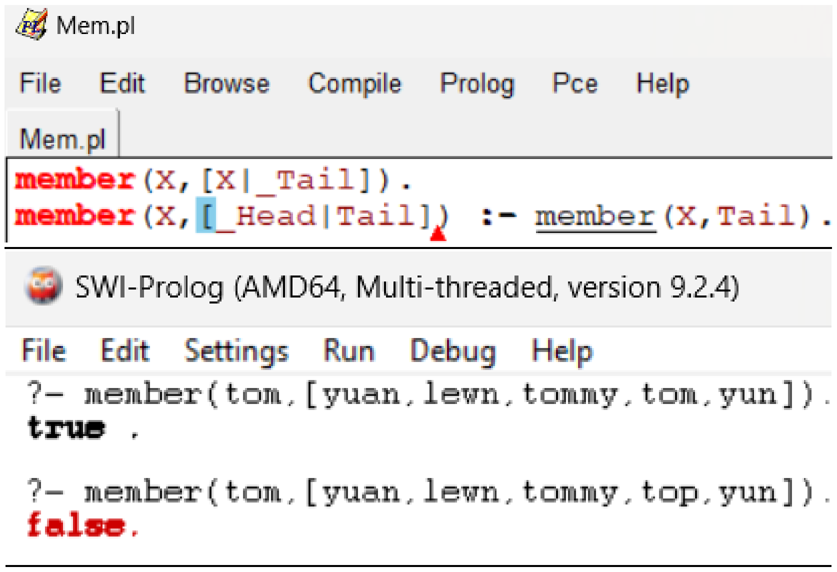
Program 7: Prolog program to implement multi (N1, N2, R) : where N1 and N2 denotes the numbers to be multiplied and R represents the result.


Multiply.pl
File Edit Browse Compile Prolog Pce Help
Multiply.pl
Multiply.pl
SWI-Prolog (AMD64, Multi-threaded, version 9.2.4)
File Edit Settings Run Debug Help

```
multi(X,Y,R) :- R is X*Y.
```

```
?- multi(8,8,R).
R = 64.
```

Program 8: Write a Prolog program to implement memb(X,L) : to check whether X is a member of L or not.

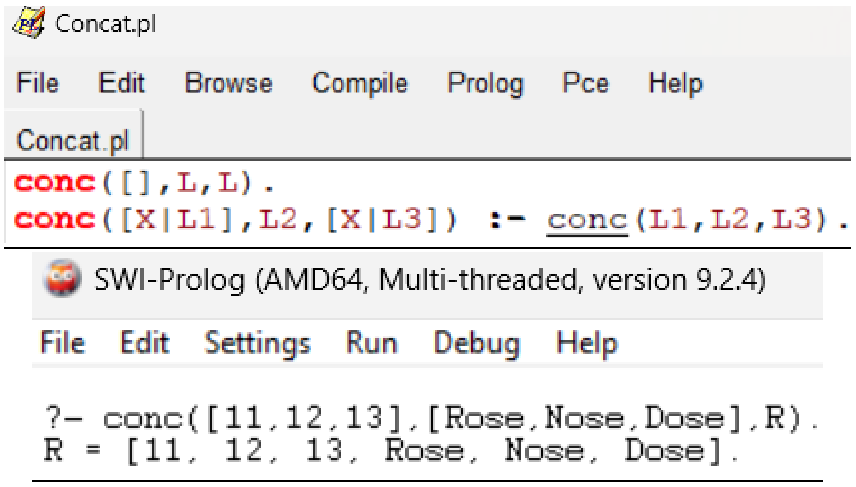

Mem.pl
File Edit Browse Compile Prolog Pce Help
Mem.pl

```
member(X,[X|_Tail]).
member(X,[_Head|Tail]) :- member(X,Tail).
```

SWI-Prolog (AMD64, Multi-threaded, version 9.2.4)
File Edit Settings Run Debug Help

```
?- member(tom,[yuan,lewn,tommy,tom,yun]).
true .

?- member(tom,[yuan,lewn,tommy,top,yun]).
false.
```

<u>Program 9: 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.</u>

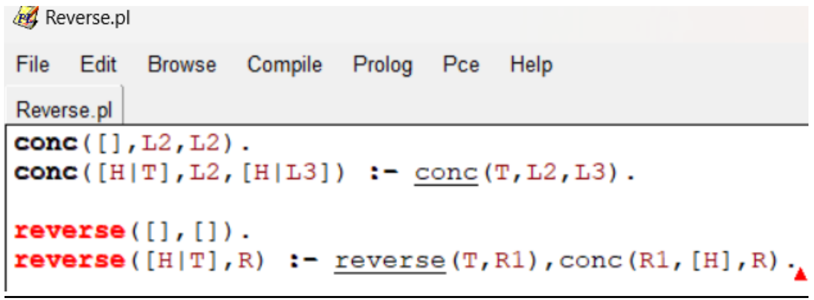

Concat.pl
File Edit Browse Compile Prolog Pce Help
Concat.pl

```
conc([],L,L).
conc([X|L1],L2,[X|L3]) :- conc(L1,L2,L3).
```

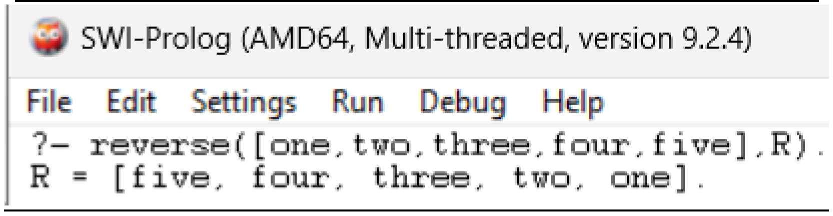
SWI-Prolog (AMD64, Multi-threaded, version 9.2.4)
File Edit Settings Run Debug Help

```
?- conc([11,12,13],[Rose,Nose,Dose],R).
R = [11, 12, 13, Rose, Nose, Dose].
```

<u>Program 10: Write a Prolog program to implement reverse(L,R) where List L is original and List R is reversed list.</u>

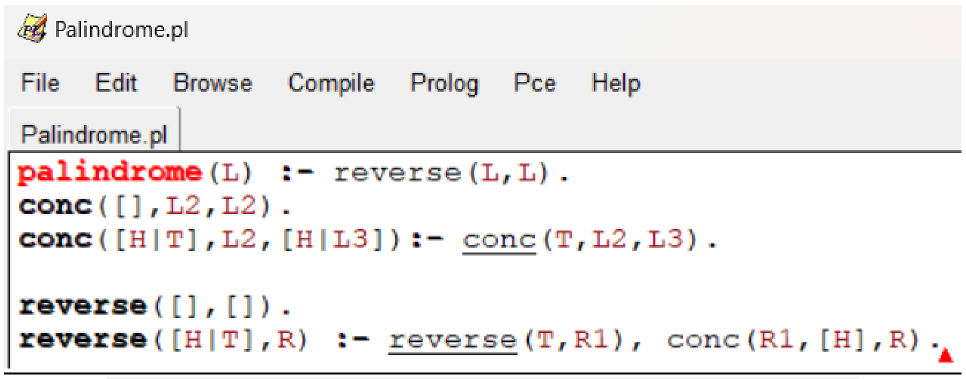

Reverse.pl

```
conc([],L2,L2).
conc([H|T],L2,[H|L3]) :- conc(T,L2,L3).

reverse([],[]).
reverse([H|T],R) :- reverse(T,R1),conc(R1,[H],R).
```

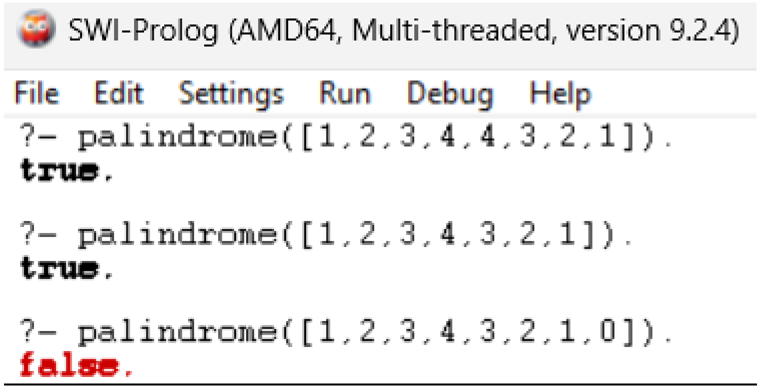

SWI-Prolog (AMD64, Multi-threaded, version 9.2.4)

```
?- reverse([one,two,three,four,five],R).
R = [five, four, three, two, one].
```

Program 11: Write a program in PROLOG to implement palindrome(L) which checks whether a list L is palindrome or not.

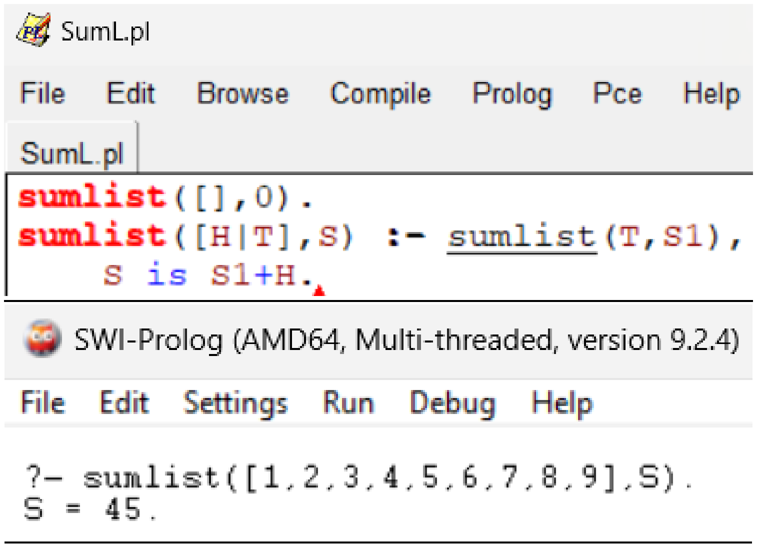

Palindrome.pl

```
palindrome(L) :- reverse(L,L).
conc([],L2,L2).
conc([H|T],L2,[H|L3]):- conc(T,L2,L3).

reverse([],[]).
reverse([H|T],R) :- reverse(T,R1), conc(R1,[H],R).
```


SWI-Prolog (AMD64, Multi-threaded, version 9.2.4)

```
?- palindrome([1,2,3,4,4,3,2,1]).
true.

?- palindrome([1,2,3,4,3,2,1]).
true.

?- palindrome([1,2,3,4,3,2,1,0]).
false.
```

Program 12: Write a Prolog program to implement sumlist(L,S) so that S is the sum of a given list L.

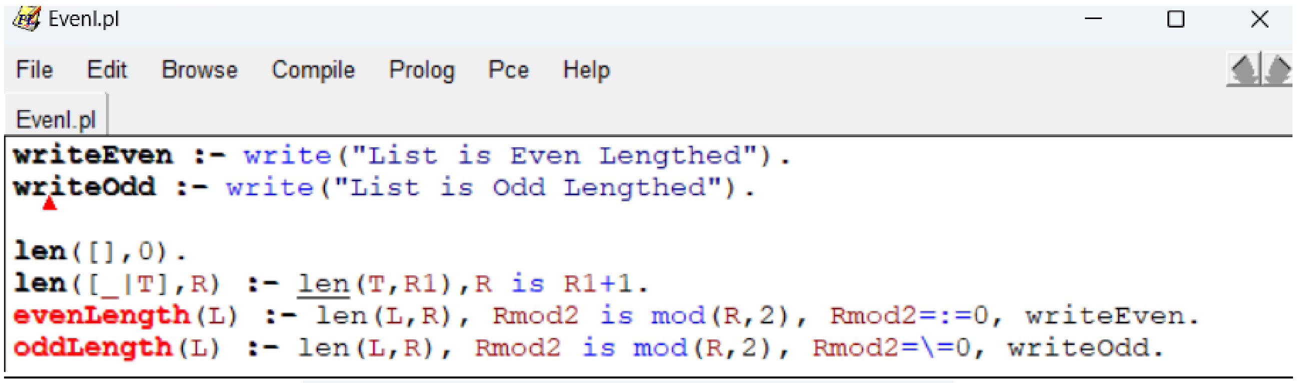

SumL.pl
File Edit Browse Compile Prolog Pce Help
SumL.pl

```
sumlist([],0).
sumlist([H|T],S) :- sumlist(T,S1),
    S is S1+H.
```

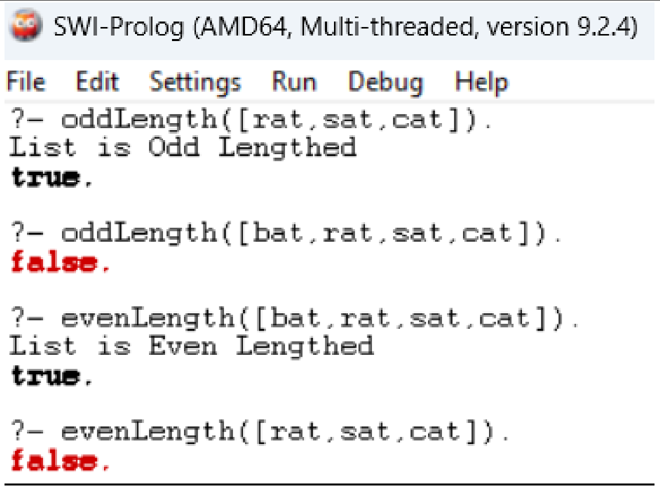
SWI-Prolog (AMD64, Multi-threaded, version 9.2.4)
File Edit Settings Run Debug Help

```
?- sumlist([1,2,3,4,5,6,7,8,9],S).
S = 45.
```

Program 13: 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.

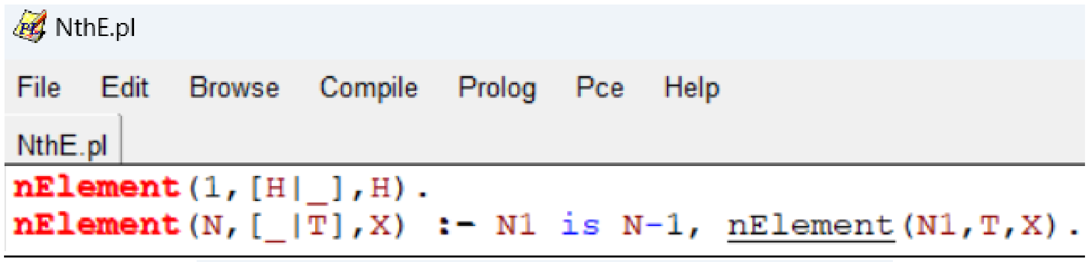


```
writeEven :- write("List is Even Lengthed").
writeOdd :- write("List is Odd Lengthed").

len([],0).
len([_|T],R) :- len(T,R1),R is R1+1.
evenLength(L) :- len(L,R), Rmod2 is mod(R,2), Rmod2=:=0, writeEven.
oddLength(L) :- len(L,R), Rmod2 is mod(R,2), Rmod2=\=0, writeOdd.
```

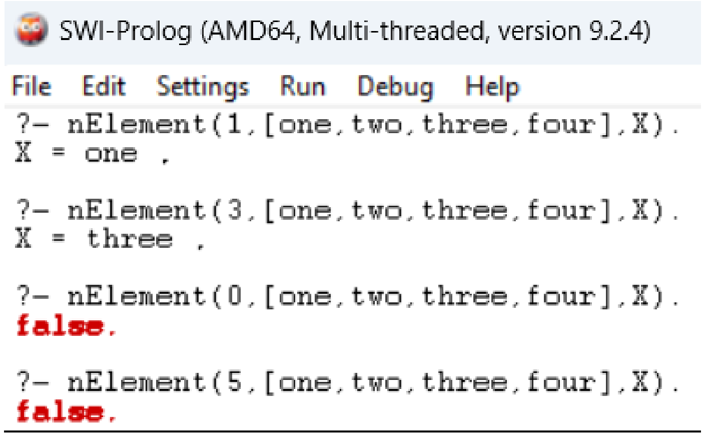

SWI-Prolog (AMD64, Multi-threaded, version 9.2.4)
File Edit Settings Run Debug Help

```
?- oddLength([rat,sat,cat]).
List is Odd Lengthed
true.

?- oddLength([bat,rat,sat,cat]).
false.

?- evenLength([bat,rat,sat,cat]).
List is Even Lengthed
true.

?- evenLength([rat,sat,cat]).
false.
```

Program 14: 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.

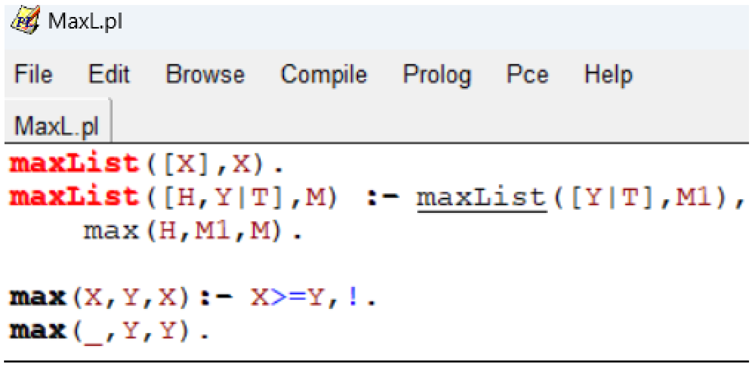


```
nElement(1,[H|_],H).
nElement(N,[_|T],X) :- N1 is N-1, nElement(N1,T,X).
```

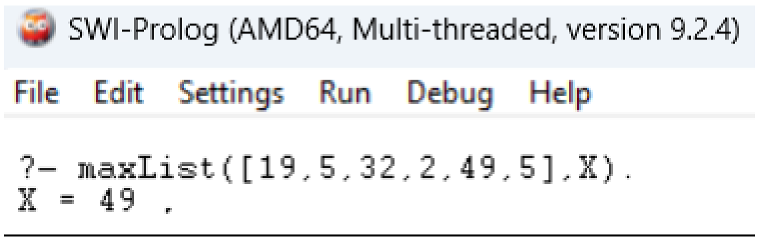

SWI-Prolog (AMD64, Multi-threaded, version 9.2.4)
File Edit Settings Run Debug Help

```
?- nElement(1,[one,two,three,four],X).
X = one .

?- nElement(3,[one,two,three,four],X).
X = three .

?- nElement(0,[one,two,three,four],X).
false.

?- nElement(5,[one,two,three,four],X).
false.
```

Program 15: Write a Prolog program to implement maxlist(L,M) so that M is the maximum number in the list.

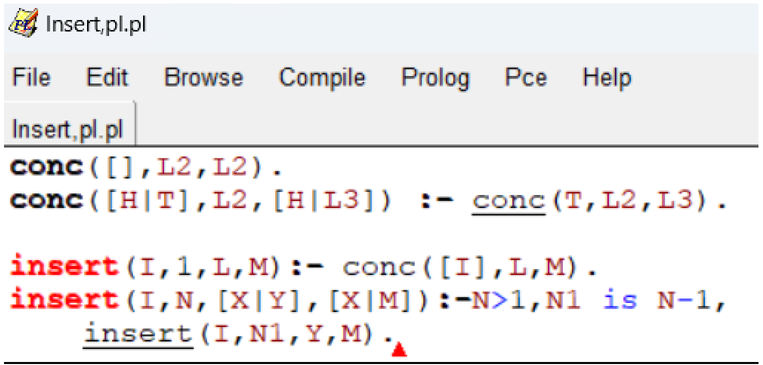


```
maxList([X],X).
maxList([H,Y|T],M) :- maxList([Y|T],M1),
    max(H,M1,M).

max(X,Y,X):- X>=Y,!.
max(_,Y,Y).
```

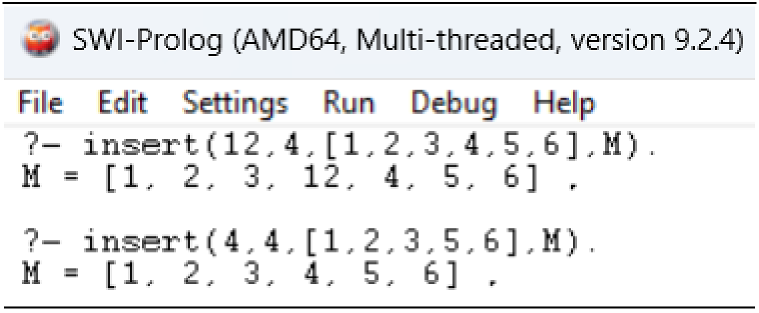

SWI-Prolog (AMD64, Multi-threaded, version 9.2.4)
File Edit Settings Run Debug Help

```
?- maxList([19,5,32,2,49,5],X).
X = 49 .
```

Program 16: 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.

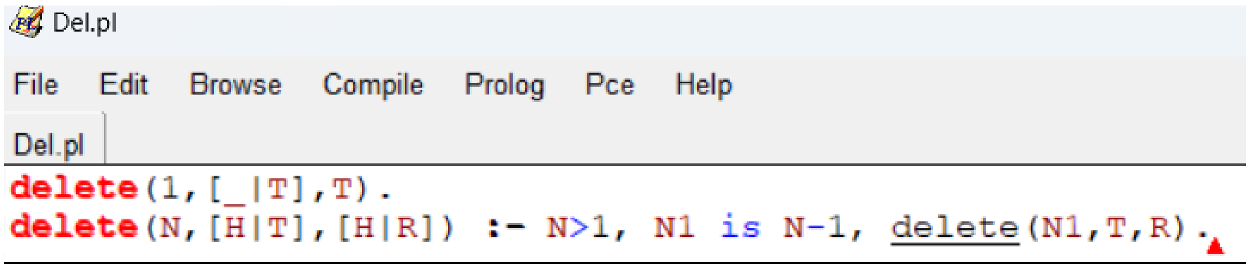


```
conc([],L2,L2).
conc([H|T],L2,[H|L3]) :- conc(T,L2,L3).

insert(I,1,L,M):- conc([I],L,M).
insert(I,N,[X|Y],[X|M]):-N>1,N1 is N-1,
    insert(I,N1,Y,M).
```

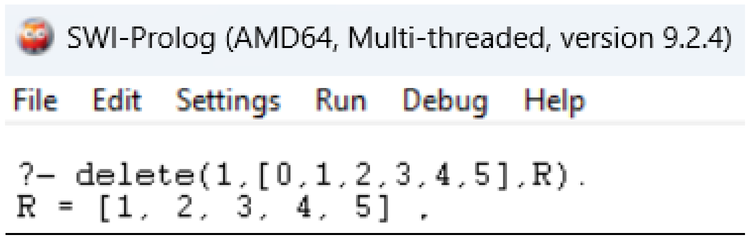

SWI-Prolog (AMD64, Multi-threaded, version 9.2.4)
File Edit Settings Run Debug Help

```
?- insert(12,4,[1,2,3,4,5,6],M).
M = [1, 2, 3, 12, 4, 5, 6] .

?- insert(4,4,[1,2,3,5,6],M).
M = [1, 2, 3, 4, 5, 6] .
```

Program 17: 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.

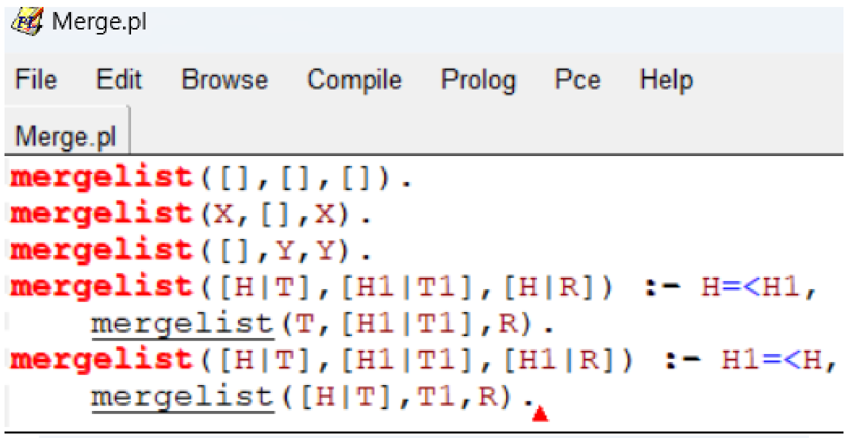


```
delete(1,[_|T],T).
delete(N,[H|T],[H|R]) :- N>1, N1 is N-1, delete(N1,T,R).
```

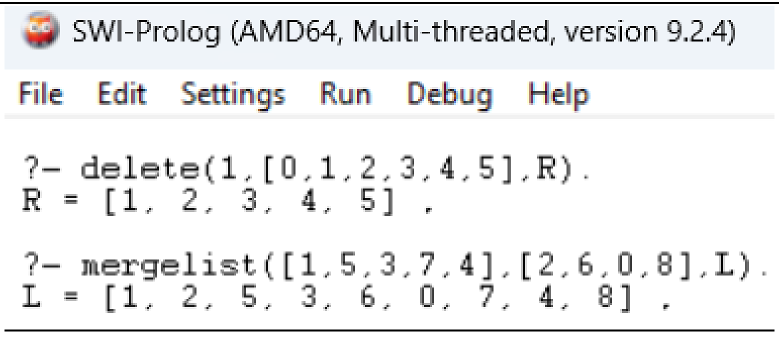

SWI-Prolog (AMD64, Multi-threaded, version 9.2.4)
File Edit Settings Run Debug Help

```
?- delete(1,[0,1,2,3,4,5],R).
R = [1, 2, 3, 4, 5] .
```

Program 18: 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.



```
mergelist([],[],[]).
mergelist(X,[],X).
mergelist([],Y,Y).
mergelist([H|T],[H1|T1],[H|R]) :- H=<H1,
    mergelist(T,[H1|T1],R).
mergelist([H|T],[H1|T1],[H1|R]) :- H1=<H,
    mergelist([H|T],T1,R).
```


SWI-Prolog (AMD64, Multi-threaded, version 9.2.4)
File Edit Settings Run Debug Help

```
?- delete(1,[0,1,2,3,4,5],R).
R = [1, 2, 3, 4, 5] .

?- mergelist([1,5,3,7,4],[2,6,0,8],L).
L = [1, 2, 5, 3, 6, 0, 7, 4, 8] .
```