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

1.pl

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

SWI-Prolog (AMD64, Multi-threaded, version 8.4.2)

File Edit Settings Run Debug Help

```
Welcome to SWI-Prolog (threaded, 64 bits, version 8.4.2)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free soft
ware.
Please run ?- license. for legal details.

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

?- sum(10,5,S).
S = 15.
```

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

2.pl

```
max(X,Y,M):-X>Y,
    M is X.
max( _,Y,M):- M is Y.
```

```
SWI-Prolog (AMD64, Multi-threaded, version 8.4.2)
File  Edit  Settings  Run  Debug  Help
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free soft
ware.
Please run ?- license. for legal details.

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

?- max(25,30,M).
M = 30.

?- max(10,2,M).
M = 10
```

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

```
3.pl
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 8.4.2)
File  Edit  Settings  Run  Debug  Help
For online help and background, visit https://www.swi-prolog.or
g
For built-in help, use ?- help(Topic). or ?- apropos(Word).

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

?- fac(10,F).
F = 3628800
```

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

4.pl

```
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 8.4.2)

File Edit Settings Run Debug Help

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

?- generate_fib(10,T).
T = 89
```

Program 5: Write a Prolog program to implement GCD of two numbers.

5.pl

```
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 8.4.2)

File Edit Settings Run Debug Help

```
Please run ?- license. for legal details.

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

?- gcd(20,25,D).
D = 5 .

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

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

6.pl

```
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.
```

SWI-Prolog (AMD64, Multi-threaded, version 8.4.2)

File Edit Settings Run Debug Help

```
Action? .

?- power(3,3,A).
A = 27
```

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.**

6.pl [modified]

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

```
|
|   multi(4,6,R).
R = 24.
```

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

8.pl

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

```
member(tom,[ann, tom, bob, pam, fin]).
true
?- member(tom,[ann, tom, bob, pam, fin]).
```

Question 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.

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

```
SWI-Prolog (AMD64, Multi-threaded, version 8.4.2)
File  Edit  Settings  Run  Debug  Help

?- conc([1,2,3],[cat,banana,rice],R).
R = [1, 2, 3, cat, banana, rice].
```

Question 10:- Write a Prolog program to implement reverse(L,R) where List L is original and List R is reversed list.

```
10.pl [modified]
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).
```

```
reverse([cat, dog, mouse ,pig, horse],R).
R = [horse, pig, mouse, dog, cat]
```

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

```
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).
```

```
palindrome([1,2,3,2,1]).
```

true

```
palindrome([1,2,3,1]).
```

false

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

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

```
sumlist([1,2,3,4,5,6,7],S).
```

S = 28

Question 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.
```

```
evenLength([cat,mouse,dog]).
false

oddLength([cat,mouse,dog]).
List is Odd-Lengthed
true

oddLength([cat,mouse,dog,pig]).
false

evenLength([cat,mouse,dog,pig]).
List is Even Lengthed
true
```

Question 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).
```

```
nElement(2,[cat,mouse,dog,pig],X).
X = mouse
```

Question 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).
```

```
maxlist([10,9,2,1,15,8],X)
X = 15
```

Question 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).
```

```
insert(23,5,[1,2,53,6,34,7],M)
M = [1, 2, 53, 6, 23, 34, 7]
```

Question 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).
```

```
delete(2,[1,2,3,5,6,7],R).
R = [1, 3, 5, 6, 7]
```

Question 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.pl

```
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).
```

```
?- mergelist([1,3,5,7,9],[2,4,6,8],L).
L = [1, 2, 3, 4, 5, 6, 7, 8, 9] .
```