Atma Ram Sanatan Dharma College

University of Delhi

Artificial Intelligence

Practical File

Submitted By -

Anshul Verma
Roll No. – 19/78065
B.Sc. (Hons.) Computer Science

Submitted To -

Dr. Parul Jain
Department of Computer Science

1. Write a Prolog program to calculate the sum of two numbers.

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

Output:

```
?- sum(4, 5, S).
S = 9.

?- sum(38, 29, X).
X = 67.
```

---

2. 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, M is X, !.
max(X, Y, M):- Y >= X, M is Y.
```

Output:

```
?- max(5, 2, M).
M = 5.

?- max(5, 18, M).
M = 18.

?- max(-37, -19, M).
M = -19.
```

---

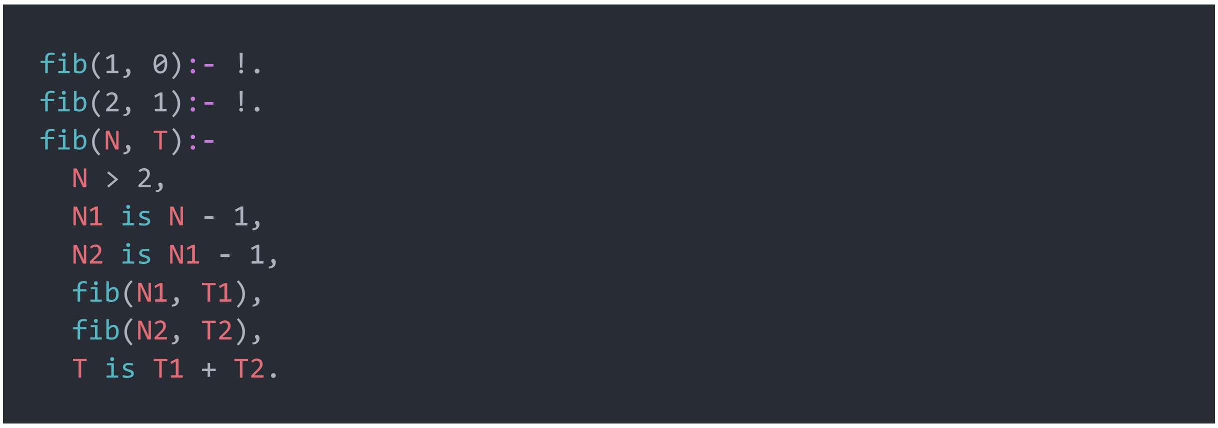
3. Write a program in PROLOG to implement factorial(N, F) where F represents the factorial of a number N.

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

Output:

```
?- factorial(1, F).
F = 1.

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

?- factorial(-5, F).
false.

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

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

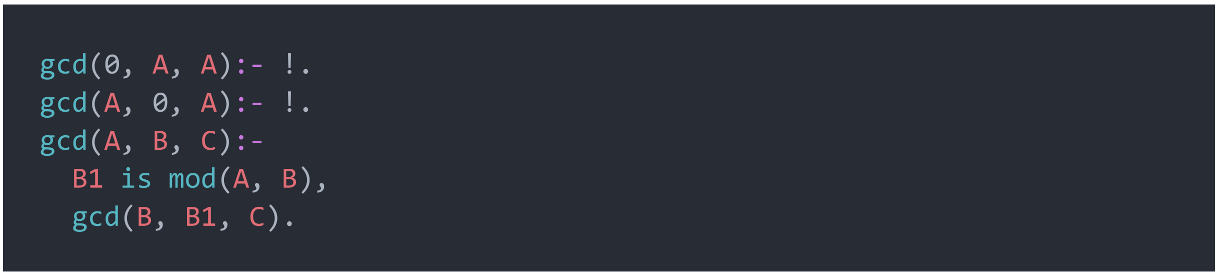


```
fib(1, 0):- !.
fib(2, 1):- !.
fib(N, T):-
  N > 2,
  N1 is N - 1,
  N2 is N1 - 1,
  fib(N1, T1),
  fib(N2, T2),
  T is T1 + T2.
```

Output:

```
 ?- fib(1, T).
  T = 0.

  ?- fib(2, T).
  T = 1.

  ?- fib(4, T).
  T = 2.

  ?- fib(10, T).
  T = 34.

  ?- fib(-1, T).
  false.
```

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

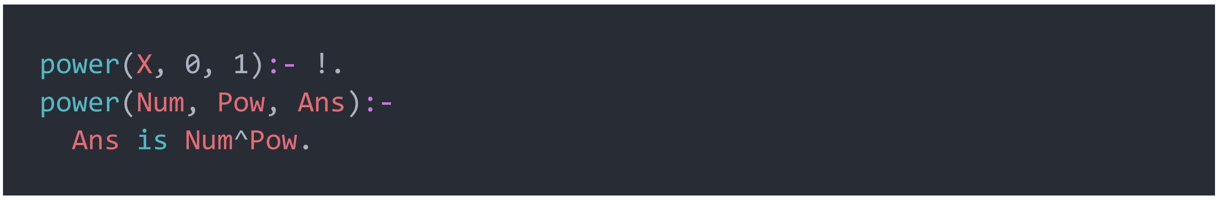


```
gcd(0, A, A):- !.
gcd(A, 0, A):- !.
gcd(A, B, C):-
  B1 is mod(A, B),
  gcd(B, B1, C).
```

Output:

```
?- gcd(15, 25, C).
C = 5.

?- gcd(0, 25, C).
C = 25.

?- gcd(12, 0, C).
C = 12.

?- gcd(12, 13, C).
C = 1.
```

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



```
power(X, 0, 1):- !.
power(Num, Pow, Ans):-
  Ans is Num^Pow.
```

Output:

```
?- power(10, 3, Ans).
Ans = 1000.

?- power(5, 6, Ans).
Ans = 15625.

?- power(11, 0, Ans).
Ans = 1.

?- power(11, -3, Ans).
Ans = 0.0007513148009015778.
```

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

```prolog
multi(N1, N2, R):-
  R is N1 * N2.
```

Output:

```
?- multi(11, 22, R).
R = 242.

?- multi(7, 15, R).
R = 105.

?- multi(7, 0, R).
R = 0.

?- multi(8, -21, R).
R = -168.
```

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

```prolog
memb(X, [X | Tail]).
memb(X, [Head | Tail]):-
  memb(X, Tail).
```

Output:

```
?- memb(b, [a, b, c]).
true .

?- memb(X, [a, b, c]).
X = a ;
X = b ;
X = c ;
false.
```

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.

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

Output:

```
?- conc([a, b, c], [1, 2, 3], L).
L = [a, b, c, 1, 2, 3].

?- conc([a, [b, c], d], [a, [], b], L).
L = [a, [b, c], d, a, [], b].

?- conc(L1, L2, [a, b, c]).
L1 = [],
L2 = [a, b, c] ;
L1 = [a],
L2 = [b, c] ;
L1 = [a, b],
L2 = [c] ;
L1 = [a, b, c],
L2 = [] ;
false.
```

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

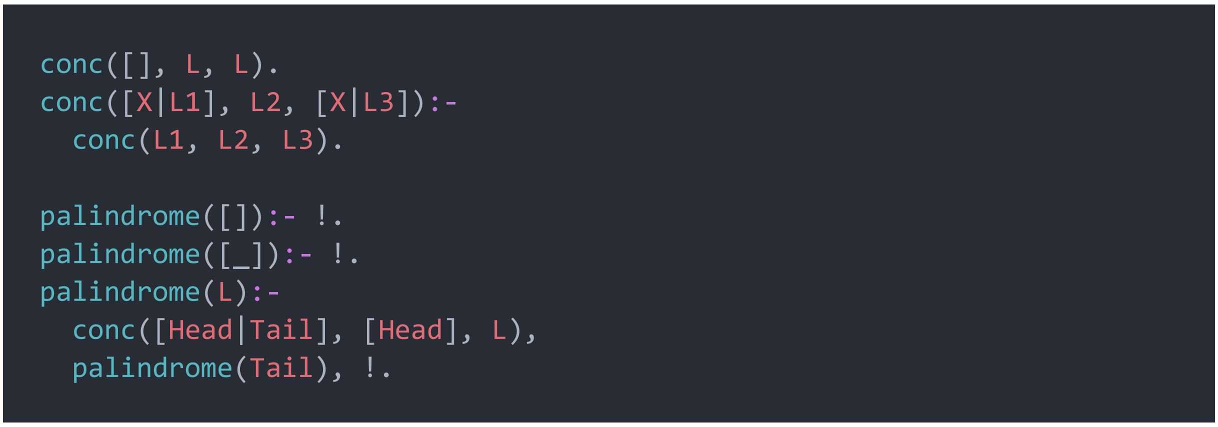
```
conc([], L, L).
conc([X|L1], L2, [X|L3]):-
  conc(L1, L2, L3).

reverse([], []).
reverse([Head|Tail], R):-
  reverse(Tail, L1),
  conc(L1, [Head], R).
```

Output:

```
?- reverse([], R).
R = [].

?- reverse([a, b, c], R).
R = [c, b, a].

?- reverse([a, [b, d], c], R).
R = [c, [b, d], a].
```

---

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



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

palindrome([]):- !.
palindrome([_]):- !.
palindrome(L):-
  conc([Head|Tail], [Head], L),
  palindrome(Tail), !.
```

Output:

```
?- palindrome([]).
true.

?- palindrome([a]).
true.

?- palindrome([a, b, a]).
true.

?- palindrome([a, b, b]).
false.
```

---

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

```
sumList([], 0).
sumList([Head|Tail], S):-
  sumList(Tail, X),
  S is Head + X.
```

Output:

```
?- sumList([1], S).
S = 1.

?- sumList([1, 2, 3], S).
S = 6.

?- sumList([], S).
S = 0.
```

---

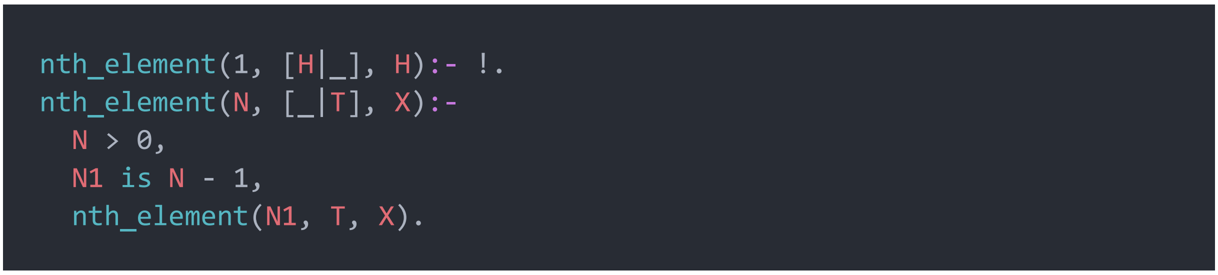
13. Write a Prolog program to implement two predicates evenlength(List) and oddlength(List) so that they are true if their argument is a list of even or odd length respectively.

```
evenlength([]):- !.
evenlength([_|T]):- oddlength(T).

oddlength([_]):- !.
oddlength([_|T]):- evenlength(T).
```

Output:

```
?- evenlength([]).
true.

?- oddlength([1]).
true.

?- oddlength([1, 2, 3, 4]).
false.

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

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.

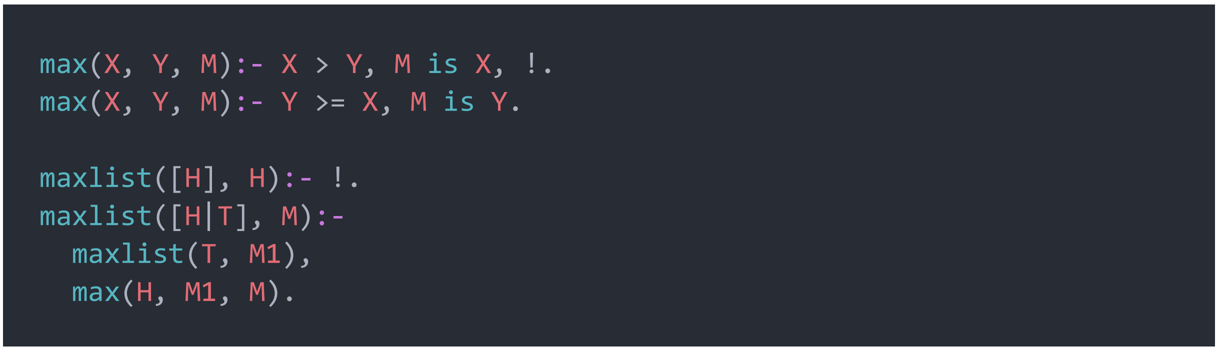


```prolog
nth_element(1, [H|_], H):- !.
nth_element(N, [_|T], X):-
  N > 0,
  N1 is N - 1,
  nth_element(N1, T, X).
```

Output:

```
?- nth_element(1, [a, b, c, d, e, f], X).
X = a.

?- nth_element(2, [a, b, c, d, e, f], X).
X = b.

?- nth_element(3, [a, b, c, d, e, f], X).
X = c.

?- nth_element(4, [a, b, c, d, e, f], X).
X = d.
```

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



```prolog
max(X, Y, M):- X > Y, M is X, !.
max(X, Y, M):- Y >= X, M is Y.

maxlist([H], H):- !.
maxlist([H|T], M):-
  maxlist(T, M1),
  max(H, M1, M).
```

Output:

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

?- maxlist([1], M).
```

```
M = 1.

?- maxlist([], M).
false.

?- maxlist([62, 37, 13, 37, 23, 82, 28], M).
M = 82.
```

---

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.

```
insert_nth(I, 1, L, [I|L]):- !.
insert_nth(I, N, [H|T], [H|T1]):-
  N1 is N - 1,
  insert_nth(I, N1, T, T1).
```

Output:

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

?- insert_nth(20, 1, [1,3,4,5], R).
R = [20, 1, 3, 4, 5].

?- insert_nth(20, 5, [23, 535, 55, 34, 56, 778, 67, 97], R).
R = [23, 535, 55, 34, 20, 56, 778, 67, 97].

?- insert_nth(25, 15, [23, 535, 55, 34, 56, 778, 67, 97], R).
false.
```

---

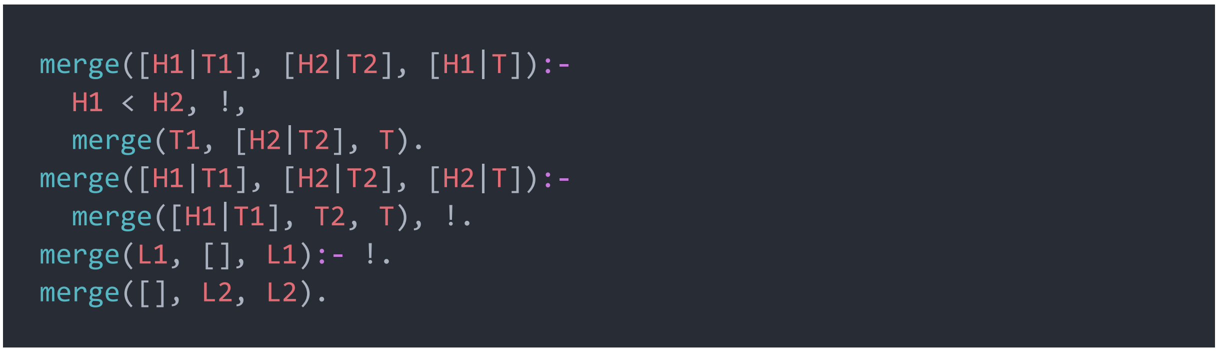
17. Write a Prolog program to implement delete_nth(N, L, R) that removes the element on Nth position from a list L to generate a list R..

```
delete_nth(1, [H|T], T):- !.
delete_nth(N, [H|T], [H|T1]):-
  N1 is N - 1,
  delete_nth(N1, T, T1).
```

Output:

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

?- delete_nth(1, [20, 1, 3, 4, 5], R).
R = [1, 3, 4, 5].

?- delete_nth(5, [23, 535, 55, 34, 20, 56, 778, 67, 97], R).
R = [23, 535, 55, 34, 56, 778, 67, 97].

?- delete_nth(15, [23, 535, 55, 34, 20, 56, 778, 67, 97], R).
false.
```

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..



```
merge([H1|T1], [H2|T2], [H1|T]):-
  H1 < H2, !,
  merge(T1, [H2|T2], T).
merge([H1|T1], [H2|T2], [H2|T]):-
  merge([H1|T1], T2, T), !.
merge(L1, [], L1):- !.
merge([], L2, L2).
```

Output:

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

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