PRACTICAL FILE

Name : Ujjawal kumar

College roll no. :20201441

Exam roll no. :20020570038

Subject : Artificial Intelligence

Course : BSc. (H) Computer Science

Semester : 6th Sem, 3rd Year

Submitted to :Dr. Subodh Sir

Date :30thApril,2023

(Ramanujan College)

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

Code

practical1.pl

```
% Prolog program to calculate the sum of two numbers

sum(X,Y) :- S is X+Y,write(S).
```

Output

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

?-
% c:/Users/Ujjawal kumar/Documents/Prolog/Practical code/practical1.pl compiled 0.00 sec, 1
 clauses
?- sum(10,5).
15
true.

?- sum(2.1,3.5).
5.6
true.

?-
```

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

Code

practical2.pl

```
% prolog program to find the maximum of two numbers

max(X,Y) :-
        X=Y -> write("both are equal");
        X>Y -> write(X);
        X<Y -> write(Y).
```

Output

```
?-
% c:/Users/Ujjawal kumar/Documents/Prolog/Practical code/practical2.pl compiled 0.00 sec, 1
 clauses
?- max(10,2).
10
true.

?- max(1,3).
3
true.

?- max(1,1).
both are equal
true.

?-
```

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

Code

practical3.pl

```
% Prolog program to find the factorial of the given number


fact(0,1).
fact(N,R) :-
        N>0 ->
                N1 is N-1,
                fact(N1,R1),
                R is N*R1;
        N<0 ->
                N1 is N+1,
                fact(N1,R1),
                R is N*R1.
```

Output

```
?- fact(5,R).
R = 120 .

?- fact(2,R).
R = 2 .

?-
```

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

Code

practical4.pl

```
% prolog program to calculate the nth fibonacci number.

fibo(1,0,1).
fibo(N,R0,R1) :-
        N1 is N-1,
        fibo(N1,R00,R01),
        R1 is R00+R01,
        R0 is R01.
```

Output

```
?- fibo(2,R0,R1).
R0 = R1, R1 = 1 .

?- fibo(7,R0,R1).
R0 = 8,
R1 = 13 .

?- fibo(12,R0,R1).
R0 = 89,
R1 = 144 .

?- fibo(3,R0,9).
```

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

Code

5.pl

```
% program to implement gcd of two number.

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).
```

Output

```
?-
% c:/Users/Ujjaval kumar/Documents/Prolog/5.pl compiled 0.00 sec, 3 clauses
?- gcd(10,20,D).
D = 10 .

?- gcd(40,15,D).
D = 5 .

?-
```

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

Code

practical6.pl

```
/**
 * 6. Prolog program to implement power(Num,Pow, Ans)
 *       : Where Num is raised to the power Pow to get Ans.
*/

power(0, Power, 0) :- Power > 0.     % Base Case
power(Num, 0, 1) :- Num > 0.         % Base Case
power(Num, Power, Ans) :- Num > 0, Power > 0,
                          P1 is Power - 1,
                          power(Num, P1, A1),
                          Ans is A1 * Num.
```

Output

```
% c:/Users/Ujjaval kumar/Documents/Prolog/Practical code/practical6.pl compiled 0.02 sec, 4
 clauses
?- power(3,3,Ans).
Ans = 27 .

?- power(2,6,Ans).
Ans = 64 .

?-
```

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

Code

practical7.pl

```
/**
 * 7. Prolog program to implement multi(N1, N2, R)
 *       : Where N1 and N2 denotes the numbers to be multiplied and R repre
sents the result.
*/

multi(N, 1, N).

multi(N1, N2, R) :- T is N2 - 1,
                    multi(N1, T, Y),
                    R is Y + N1.
```

Output

```
% c:/Users/Ujjawal kumar/Documents/Prolog/Practical code/practical7.pl compiled 0.02 sec, 3
 clauses
?- multi(3,6,R).
R = 18 .

?- multi(2,9,R).
R = 18 .

?-
```

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

Code

practical8.pl

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

mem(X, [X, _]) :- !.     % Base Case
mem(X, [_ | L]) :- mem(X, L).    % Recursive Case
```

Output

```
?- mem(tom,[ant,cat,tom,dog]).
true.

?-
```

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.

Code

practical9.pl

```
/**
 * 9.  Write a Prolog program to implement conc(L1, L2, L3),
 *          : where L2 is the list to be appended with L1 to get the resul
ted list L3.
 *
*/

conc([], List, List).                                   % Base Call
conc([Head | Tail], List2, [Head | Result]) :-
    conc(Tail, List2, Result).                          % Recursive Call
```

Output

```
% c:/Users/Ujjawal kumar/Documents/Prolog/Practical code/practical9.pl compiled 0.00 sec, 3
 clauses
?- conc([11,12,19,21],[aam,cat,home],Result).
Result = [11, 12, 19, 21, aam, cat, home].

?-
```

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

Code

practical10.pl

```
/**
 * 10. Write a Prolog program to implement reverse(L, R),
 *         : Where List L is original and List R is reversed list.
 *
 *
*/

append([], L, L).                              % Base Call
append([X|L1], L2, [X|L3]) :-
  append(L1, L2, L3).                        % Recursive Call

reverse([], []).                               % Base Call
reverse([H|T], R) :-
  reverse(T, L1),                            % Recursive Call
  append(L1, [H], R).
```

Output

```
% c:/Users/Ujjawal kumar/Documents/Prolog/Practical code/practical10.pl compiled 0.02 sec,
5 clauses
?- reverse([apple,mango,pig,home],R).
R = [home, pig, mango, apple].

?-
```

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

Code

practical11_a.pl

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

palindrome([]) :-
  write('Palindrome!').

palindrome([_]) :-
  write('Palindrome!').

palindrome(L) :-
  append([H|T], [H], L),
  palindrome(T);
  write('Non-Palindrome!').
```

practical11_b.pl

```
/**
 * 11. Write a prolog program to implement palindrome(L),
 *           : which checks whether L is palindrome or not.
 *
*/

conc([], List, List).                                  % Base Call
conc([Head | Tail], List2, [Head | Result]) :-
  conc(Tail, List2, Result).                        % Recursive Call

reverse([], []).                                  % Base Call
reverse([Head | Tail], R) :-
  reverse(Tail, RT),                           % Recursive Call
  conc(RT, [Head], R).

palindrome(L) :- reverse(L, L).
```

Output

```
?- palindrome([1,2,1]).
Palindrome!
true .

?- palindrome([wow]).
Palindrome!
true .

?-
```

```
% c:/Users/Ujjawal kumar/Documents/Prolog/Practical code/practical11_b.pl compiled 0.02 sec
, 6 clauses
?- palindrome([1,2,3,2,1]).
true.

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

?-
```

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

Code

practical12.pl

```
/**
 * 12. Write a prolog program to implement sumlist(L, S),
 *           : so that S is the sum of a given list L.
 *
*/

sumlist([], 0).                              % Base Call
sumlist([Head | Tail], S) :-
  sumlist(Tail, S1), S is Head + S1.      % Recursive Call
```

Output

```
% c:/Users/Ujjawal kumar/Documents/Prolog/Practical code/practical12.pl compiled 0.02 sec,
3 clauses
?- sumlist([1,3,2,5,8,9],S).
S = 28.

?-
```

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.

Code

practical13.pl

```
/**
 * 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 o
dd length respectively.
 */

oddlength([_| Tail]) :-
  evenlength(Tail).

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

Output

```
% c:/Users/Ujjaval kumar/Documents/Prolog/Practical code/practical13.pl compiled 0.00 sec,
4 clauses
?- oddlength([cat,3,dog]).
true.

?- oddlength([cat,dod]).
false.

?- evenlength([cat,dod]).
true.

?- evenlength([cat,dod,van]).
false.

?-
```

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.

Code

practical14.pl

```
/**
 * 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).
```

Output

```
% c:/Users/Ujjawal kumar/Documents/Prolog/Practical code/practical14.pl compiled 0.00 sec,
3 clauses
?- nElement(2,[cat,mouse,dog,pig],X)
|    .
X = mouse ,

?- nElement(3,[cat,mouse,dog,pig],X).
X = dog ,

?-
```

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

Code

practical15.pl

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

maxlist([], 0).

maxlist([Head | Tail], Max) :-
  maxlist(Tail, TailMax),
  Head > TailMax,
  Max is Head.

maxlist([Head | Tail], Max) :-
  maxlist(Tail, TailMax),
  Head =< TailMax,
  Max is TailMax.
```

Output

```
% c:/Users/Ujjawal kumar/Documents/Prolog/Practical code/practical15.pl compiled 0.00 sec,
4 clauses
?- maxlist([1,15,3,18,22,1,41,3],Max).
Max = 41 ,

?- maxlist([1,15,3,18,22,1,41,80],Max).
Max = 80 ,

?- maxlist([100,15,3,18,22,1,41,3],Max).
Max = 100 ,

?-
```

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.

Code

practical16.pl

```
/**
 * 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 genera
te a list R.
 */

insert_nth(I, 0, L, [I | L]).

insert_nth(I, N, [Head | Tail], [Head | Tail1]) :-
  N1 is N - 1,
  insert_nth(I, N1, Tail, Tail1).
```

Output

```
% c:/Users/Ujjawal kumar/Documents/Prolog/Practic
3 clauses
?- insert_nth(10,3,[1,3,14,5,8,9],Tail1).
Tail1 = [1, 3, 14, 10, 5, 8, 9] .

?- insert_nth(10,1,[1,3,14,5,8,9],Tail1).
Tail1 = [1, 10, 3, 14, 5, 8, 9] .

?- insert_nth(10,0,[1,3,14,5,8,9],Tail1).
Tail1 = [10, 1, 3, 14, 5, 8, 9] .

?-
```

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.

Code

practical17.pl

```
/**
 * 17. Write a Prolog program to implement delete_nth(N, L, R),
 *          : that removes the element on Nth position from a list L to gene
rate a list R.
 */

delete_nth(0, [_ | Tail], Tail).

delete_nth(N, [Head | Tail], [Head | Tail1]) :-
  N1 is N - 1,
  delete_nth(N1, Tail, Tail1).
```

Output

```
% c:/Users/Ujjawal kumar/Documents/Prolog/Practical code/practical17.pl
3 clauses
?- delete_nth(3,[1,5,7,3,9,13],Tail1).
Tail1 = [1, 5, 7, 9, 13] .

?-
```

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.

Code

practical18.pl

```
/**
 * 18. Write a program in PROLOG to implement merge(L1, L2, L3),
 *         : where L1 is first ordered list and L2 is second ordered list a
nd L3 represents the merged list.
 */

mergelist(L1, [], L1).
mergelist([], L2, L2).

mergelist([Head1 | Tail1], [Head2 | Tail2], L3) :-
  Head1 < Head2 -> append([Head1], L2, L3),
    mergelist(Tail1, [Head2 | Tail2], L2);

  Head1 > Head2 -> append([Head2], L2, L3),
    mergelist([Head1 | Tail1], Tail2, L2);

  append([Head1 | Head2], L2, L3),
    mergelist(Tail1, Tail2, L2).
```

Output

```
?- mergelist([1,3,4,9],[2,6,7,8],L2).
L2 = [1, 2, 3, 4, 6, 7, 8, 9] .

?-
```