ARTIFICIAL INTELLGENCE PRACTICAL FILE

NAME:- RAJ KHATRI
EXAMINAITON ROLL NUMBER:- 20001570040
PAPER:- ARTIFICIAL INTELLIGENCE (32341601)
COURSE:- B.SC. (HONS.) COMPUTER SCIENC
SUPERVISOR: PROF. VIBHA GAUR, MS. AMRITA, MR. MEHTAB ALAM

Q1. WRITE A PROLOG PROGRAM TO IMPLEMENT MULTI (N1, N2, R) WHERE N1 AND N2 DENOTE THE NUMBERS TO BE MULTIPLIED AND R REPRESENTS THE RESULT.

```
%Sum of two numbers.
q1_sum():-
    write("Enter the first number: "),
    read(N1), %Enter number 1
    write("Enter the second number: "),
    read(N2), % Enter number 2
    sum(N1,N2,R). %calling predicate

sum(N1,N2,R):-
    R is N1+N2, % storing sum of two numbers in R
    write("The sum is: "),
    write(R). % displaying results
```

```
2 ?- q1_sum().
Enter the first number: 6.
Enter the second number: |: 9.
The sum is: 15
true.
```

Q2. WRITE A PROLOG PROGRAM TO IMPLEMENT REVERSE (L, R) WHERE LIST L IS ORIGINAL AND LIST R IS REVERSED LIST.

```
q2_max():-
    write("Enter the first number: "),
    read(N1), %Enter number 1
    write("Enter the second number: "),
    read(N2), %Enter number 2
    max(N1,N2,M). %calling the predicate

max(X,Y,M):-
    %if both are equal display equal
    X=Y -> write('Both are equal');
    % if X > Y then display X
    X>=Y ->
        M is X,
        write(M);
    % if Y > X then display Y
    M is Y,
        write(M).
```

```
2 ?- q2_max().
Enter the first number: 6.
Enter the second number: |: 9.
9
true.
```

Q3. WRITE A PROGRAM IN PROLOG TO IMPLEMENT FACTORIAL (N, F) WHERE F REPRESENTS THE FACTORIAL OF A NUMBER N.

```
q3_factorial():-
    write("Enter the number: "),
    read(N1), %Enter number
    factorial(N1,F), %calling the predicate
    write(F).

factorial(0,1). % factorial of 0 is 1
factorial(1,1). % factorial of 1 is 1
factorial(N,F) :- %recursive predicate to calculate factorial
    M is N-1, %N is decremented and multiplied with result
    factorial(M,F1), %calling it till reaches the base condition
    F is N*F1.
```

```
4 ?- q3_factorial().
Enter the number: 5.
120
true .
```

Q 4. WRITE A PROGRAM IN PROLOG TO IMPLEMENT GENERATE_FIB (N, T) WHERE T REPRESENTS NTH TERM OF THE FIBONACCI SERIES.

```
q4_fibonacci():-
    write("Enter the number: "),
    read(N1), %Enter number 1
    fibonacci(N1,F), %calling the predicate
    write(F).

fibonacci(1,0).
fibonacci(2,1).
fibonacci(N,T):-
    N>1,
    N1 is N-1, %N1 is decremented till it reaches the base condition
    N2 is N-2, %N2 is decremented till it reaches the base condition
    fibonacci(N1,T1),
    fibonacci(N2,T2),
    T is T1+T2.
```

```
6 ?- q4_fibonacci().
Enter the number: 5.
3
true .
```

Q 5. WRITE A PROLOG PROGRAM TO IMPLEMENT GCD OF TWO NUMBERS.

```
q5_gcd():-
    write("Enter the first number: "),
    read(N1), %Enter number 1
    write("Enter the second number: "),
    read(N2), %Enter number 2
    gcd(N1,N2,R), %calling the predicate
    write(R).

gcd(X,0,X). %if any number 0 then resultant will be other number

gcd(X,Y,Z):- %predicate for calculating the gcd
    Res is mod(X,Y), % it is calculated using the mod operator that returns the remainder
after division.
    gcd(Y,Res,Z). % recursively gcd predicate is called.
```

```
8 ?- q5_gcd().
Enter the first number: 6.
Enter the second number: |: 9.
3
true .
```

Q 6. WRITE A PROLOG PROGRAM TO IMPLEMENT POWER(NUM, POW, ANS) WHERE NUM IS RAISED TO THE POWER POW TO GET ANS.

```
q6_power():-
    write("Enter the number: "),
    read(N), %Enter number
    write("Enter the power: "),
    read(P), %Enter power
    power(N,P,Ans), %calling predicate
    write(Ans).

power(Num, Pow, Ans):-
    Ans is Num^Pow. %performing power
```

```
10 ?- q6_power().
Enter the number: 6.
Enter the power: |: 9.
10077696
true.
```

Q 7. PROLOG PROGRAM TO IMPLEMENT MULTI(N1, N2, R) WHERE N1 AND N2 DENOTES THE NUMBERS TO BE MULTIPLIED AND R REPRESENTS THE RESULT.

```
q7_multiply():-
    write("Enter the first number: "),
    read(N1), %Enter number 1
    write("Enter the second number: "),
    read(N2), %Enter number 2
    multi(N1,N2). %calling predicate
```

```
multi(N1,N2):-
    R is N1*N2, %multiplying two numbers
    write("The product is: "),
    write(R).
```

```
12 ?- q7_multiply().
Enter the first number: 6.
Enter the second number: |: 9.
The product is: 54
true.
```

Q 8. WRITE A PROLOG PROGRAM TO IMPLEMENT MEMB(X, L) TO CHECK WHETHER X IS A MEMBER OF L OR NOT.

```
q8_member():-
    write("Enter the list: "),
    read(L), %Enter list
    write("Enter the element to check in the list: "),
    read(E), %Enter element
    memb(E,L). %calling predicate

%list contains the element at the head position
memb(X,[X|_]).

memb(X,[_|T]):-
    memb(X,T). % element is checked in the tail part
```

```
13 ?- [q8_member].
true.
Enter the element to check in the list: |: 6.

true .

15 ?- q8_member().
Enter the list: [1,2,3,4,5,6,7,8].
Enter the element to check in the list: |: 9.
```

Q 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.

```
q9_concat():-
    write("Enter the first list: "),
    read(L1), %Enter first list
    write("Enter the second list: "),
    read(L2), %Enter second list
    conc(L1,L2,L3), %calling predicate
    write(L3).

 %if the first list is empty so the second is returned
conc([],L,L).
```

```
 %predicate is defined, head is split from first and stoored in the second
conc([H1|L1],L2,[H1|L3]):-
    conc(L1,L2,L3). %called recursively
```

```
16 ?- q9_concat().
Enter the first list: [1,2,3,4,5].
Enter the second list: |: [6,7,8,9].
[1,2,3,4,5,6,7,8,9]
true.
```

Q 10. WRITE A PROLOG PROGRAM TO IMPLEMENT REVERSE(L, R) WHERE LIST L IS ORIGINAL AND LIST R IS REVERSED LIST.

```
q10_reverse():-
    write("Enter the list: "),
    read(L1), %Enter list
    reverse(L1,R), %calling predicate
    write(R).

reverse([],[]). %if the list is empty the result is empty list

reverse([H|T],R):- %the list is divied into head and tail part
    reverse(T,ReversedTail), %reverse is called on the tail
    append(ReversedTail,[H],R). %head is appended on reversed list obtained recursively.
```

```
18 ?- q10_reverse().
Enter the list: [1,2,3,4,5,6].
[6,5,4,3,2,1]
true.
```

Q 11. WRITE A PROGRAM IN PROLOG TO IMPLEMENT PALINDROME(L) WHICH CHECKS WHETHER A LIST L IS A PALINDROME OR NOT.

```
q11_palindrome():-
    write("Enter the list: "),
    read(L1), %Enter list
    palindrome(L1). %calling predicate

palindrome(L):-
    %calling reverse predicate
    %it returns true if it is same
    reverse(L,L).
```

```
20 ?- q11_palindrome().
Enter the list: [1,2,3,2,1].

true.

21 ?- q11_palindrome().
Enter the list: [6,9].

false.
```

Q 12. WRITE A PROLOG PROGRAM TO IMPLEMENT SUMLIST(L, S) SO THAT S IS THE SUM OF A GIVEN LIST L.

```
q12_sumlist():-
    write("Enter the list: "),
    read(L1), %Enter list
    sumlist(L1,R), %calling predicate
    write(R).

sumlist([],0). %if list is empty return 0

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

```
22 ?- q12_sumlist().
Enter the list: [1,2,3,4,5,6,7,8,9].
45
true.
```

Q 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.

```
q13_listlength():-
    write("Enter the list : "),
    read(L1), %Enter list
    calc_length(L1). %calling predicate

oddlength:- %predicate is called when the list odd length
    write('Odd Length').

evenlength:- %predicate is called when the list even length
    write('Even Length').

calc_length([]):- %predicate is called when the list is empty
    write('List is empty').

calc_length(L):-
    length(L,N), %length is an in-built predicate
    L1 is mod(N,2), %condition is checked for even or odd
    L1=:=0 -> %if len % 2 is 0 then evenlength is called
```

```
    evenlength;
    oddlength.
```

```
24 ?- q13_listlength().
Enter the list : [1,2,3,5].
Even Length
true.

25 ?- q13_listlength().
Enter the list : [1,2,3,4,5].
Odd Length
true.
```

Q 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.

```
q14_checkelement():-
    write("Enter the list: "),
    read(L), %Enter list
    write("Enter the position: "),
    read(P), %Enter position
    write("Enter the element: "),
    read(E), %Enter element
    nth_Element(P,L,E). %calling predicate

nth_Element(0,[H|_],H). % If element is found

nth_Element(N,[_|T],H ):-
    % Check in the tail of the list and increment the resulting index
    nth_Element(N1,T,H),
    N is N1+1.
```

```
26 ?- q14_checkelement().
Enter the list: [1,2,3,4,5,6].
Enter the position: |: 3.
Enter the element: |: 3.

false.

27 ?- q14_checkelement().
Enter the list: [1,2,3,4,5,6].
Enter the position: |: 3.
Enter the element: |: 4.

true .
```

Q 15. WRITE A PROLOG PROGRAM TO IMPLEMENT MAXLIST(L, M) SO THAT M IS THE MAXIMUM NUMBER IN THE LIST.

```
q15_maxlist():-
    write("Enter the list: "),
    read(L), %Enter list
    maxlist(L,R), %calling predicate
    write(R).

maxlist([X],X).

maxlist([X|T], M):-
    maxlist(T, M),
    M >= X. %returns tail if it is greater than the head
    maxlist([X|T], X):-
    maxlist(T, M),
    X > M.
```

```
28 ?- q15_maxlist().
Enter the list: [1,2,3,4,5,6,7,8,9].
9
true .
```

Q 16. WRITE A PROLOG PROGRAM TO IMPLEMENT INSERT_NTH(I, N, L, R) THAT INSERTS AN ITEM I INTO NTH POSITION OF A LIST L TO GENERATE A LIST R.

```
q16_insert():-
    write("Enter the list: "),
    read(L), %Enter list
    write("Enter the position: "),
    read(P), %Enter postiong
    write("Enter the element: "),
    read(E), %Enter element
    insert_nth(E,P,L,R), %calling predicate
    write(R).

insert_nth(I,1,[H1|T],[H1,I|T]).

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

```
30 ?- q16_insert().
Enter the list: [1,2,3,4,5,6].
Enter the position: |: 6.
Enter the element: |: 7.
[1,2,3,4,5,6,7]
true .
```

Q 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.

```
q17_delete():-
    write("Enter the list: "),
    read(L), %Enter lsit
```

```
    write("Enter the position to be deleted: "),
    read(P), %Enter position
    delete_nth(P,L,R), %calling predicate
    write(R).

delete_nth(1,[_|T],T). %base condition

delete_nth(P,[X|Y],[X|R]):-
    P1 is P-1, %iterating list by decrementing the position
    delete_nth(P1,Y,R).
```

```
32 ?- q17_delete().
Enter the list: [1,2,3,4,5,6,7,8,9].
Enter the position to be deleted: |: 6.
[1,2,3,4,5,7,8,9]
true .
```

Q 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.

```
q18_merge():-
    write("Enter the first ordered list: "),
    read(L1), %Enter list 1
    write("Enter the second ordered list: "),
    read(L2), %Enter list 2
    merge(L1,L2,L3), %calling predicate
    write(L3).

merge([],[],[]). %both lists are empty
merge(L,[],L). %if one list empty returing other
merge([],L,L). %if one list empty returing other

%predicate for merge where head of first is added to result
%if it is smaller than head of second list
merge([H1|T1],[H2|T2],[H1|T3]):-
    H1=<H2,
     %recursion for rest elements
    merge(T1,[H2|T2],T3).

%else head of second list is added first to resultant list
merge([H1|T1],[H2|T2],[H2|T3]):-
    merge([H1|T1],T2,T3).
```

```
34 ?- q18_merge().
Enter the first ordered list: [1,3,5,7].
Enter the second ordered list: |: [2,4,6,8].
[1,2,3,4,5,6,7,8]
true
```