JNIESTRT'S

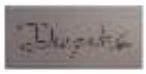
SMT. INDIRA GANDHI COLLEGE OF ENGINEERING

GHANSOLI, NAVI MUMBAI – 400701

(Approved by AICTE New Delhi & Govt. of Maharashtra, Affiliated to University of Mumbai)

COMPUTER ENGINEERING DEPARTMENT

ACADEMIC YEAR :- 2021-22(EVEN SEM)



NAME- DEEPAK H CHOURASIYA
ROLL NO - 77
YEAR - TE SEM - VI
BRANCH - COMPUTER

LAB NO : 06

TITLE-: INTRODUCTION TO PROLOG PROGRAMMING AND BASIC PROGRAMMING IN PROLOG

Date of Performance	Date of Evaluation	Marks (10)					Sign / Remark
		A	B	C	D	E	
		2	3	2	2	1	
01/03/22	08/03/22						
		Total Marks					

Date:

Date	Experiment	Grade:
01-03-22	Exp-6: Introduction to prolog programming and basic programming in prolog	Sign:

Aim: To implement basic programs in prolog.

Theory: Prolog is also called as programming in logic.

- Prolog has the ability to infer facts from the given facts and rules.
- It has important role in artificial intelligence unlike many other languages.
- Stack scheduling policy is adopted. It maintains the resolvent as a stack. It pops the top subgoal for reduction and pushes the derived goals on the resolvent stack.

Components of Prolog:

1. Constants: Numericals & symbols such as 4, many, etc.

Date:_____

2. String : String of characters within single quote.

3. Function and Predicate names :
start with alphabet and formed by using lower case letters, numericals and underscore (_).

4. Variable names are formed similar to function and predicate but it must start with upper case letter. Normally we use X, Y, Z .... for variables.

5. Clause : (fact or Rule) is terminated by full stop (.).

6. Goal to a prolog program is given after symbols ?-

Prolog control strategy uses Depth first traversal, it contains three control strategies.

I. Forward Movement :

Choose rule by -

- Searching sequentially in the program from top to bottom whose head matches with the goal with possible unifier.

Date:

- Remember the position of the matched rule.
- Join the rule body in front of the sequence of sub goals to be solved.

II. Unification: It is the process of matching or finding the most general unifier.

III. Backtracking: When a task fails, prolog traces backwards & tries to satisfy previous task.

a) Shallow Backtracking:

It occurs when the last sub-goal succeeds. To find alternative solution, the last subgoal is tried to be unified with another rule (if any) from last place marker.

b) Deep Backtracking:

It occurs as a result of their failure as shallow in which it backtracks to previous sub-goal. Remove bindings generated by sub goals introduced due to current subgoal. Look for an alternative rule (with the head matching goal, if any) after the place marker for previous sub goal.

Date:_____

Prolog Programs:

Program 1: Computing square of a number.

Predicates : squareOf (integer, integer)

Clauses :

```
squareOf(X,Y) :- Y is X*X.
```

Goal :

```
?- squareOf(2,Y)

Y=4

?- squareOf(X,9)   { fails :: left hand side has to be variable }

?- squareOf(3,9)
True.
```

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 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).

?-
% d:/prolog/square_of_number compiled 0.00 sec, -2 clauses
% d:/prolog/square_of_number compiled 0.00 sec, 0 clauses
?- squareOf(0,Answer).
Answer = 0.

?- squareOf(1, Answer).
Answer = 1.

?- squareOf(2, Answer).
Answer = 4.

?- squareOf(10, Answer).
Answer = 100.

?- squareOf(15, Answer).
Answer = 225.

?- squareOf(50, Answer).
Answer = 2500.

?- squareOf(100, Answer).
Answer = 10000.

?- squareOf(-5, Answer).
Answer = 25.

?-
```

square_of_number.pl

File Edit Browse Compile Prolog Pce Help

square_of_number.pl

```
%Computing square of a number
squareOf(X, Y):-Y is X*X.
```

Date:_____

Program 2: Write a ~~prolog program~~ to check whether an element of is a member of a given list or not.

```
% include domains pro
domains
        charlist = char*
        integerlist = integer*
        namelist = symbol*

Predicates:
        member (char, charlist)

Clauses:
        member (X, [X|_]).
        member (X, [_|T]) :- member (X, T).

?- member (d, [a,b,c,d,e,f])
   True.

?- member (d, [a,b,c,d,e,f])
(2) ↓
   member (d, [b,c,d,e,f])
(2) ↓
   member (d, [c,d,e,f])
(2) ↓
   member (d, [d,e,f])
(1) ↓
   Succeeds (True)
```

Date:_____

?- member(d, [a, b, c])

② ↓

member(d, [b, c])

② ↓

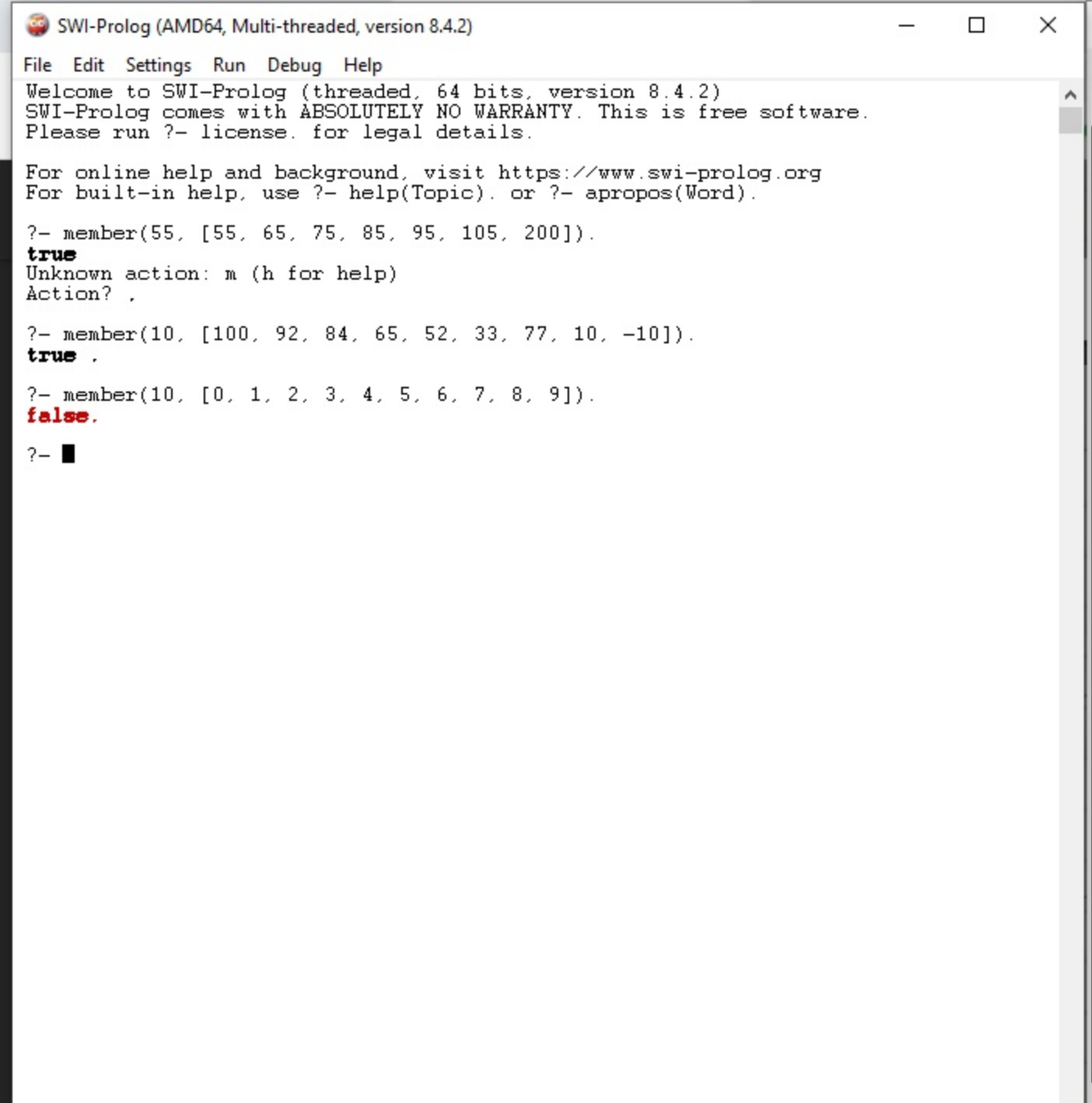
member(d, [c])

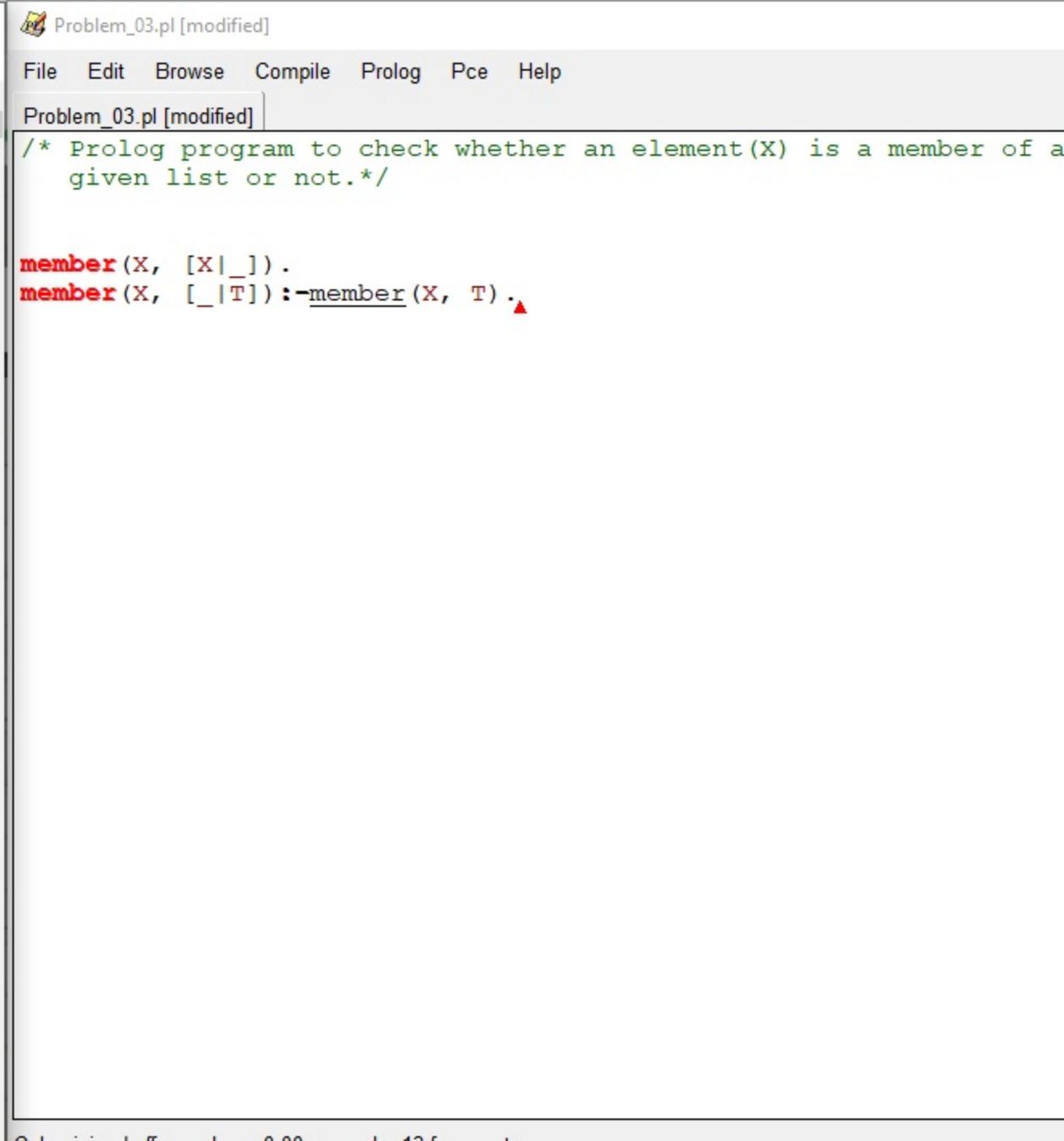
② ↓

member(d, [])

↓

fail (false)


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 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).
?- member(55, [55, 65, 75, 85, 95, 105, 200]).
true
Unknown action: m (h for help)
Action? .
?- member(10, [100, 92, 84, 65, 52, 33, 77, 10, -10]).
true .
?- member(10, [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]).
false.
?-


Problem_03.pl [modified]
File Edit Browse Compile Prolog Pce Help
Problem_03.pl [modified]
/* Prolog program to check whether an element(X) is a member of a
given list or not.*/
member(X, [X|_]).
member(X, [_|T]):-member(X, T).

Date:_____

Program 3: To find the length of the list.

Length (L) = { 0 if L [ ]
1 + length (tail (L)), otherwise

domains:
charlist = char *
integerlist = integer *

Predicates:
lengthOfList ([ ], 0).
lengthOfList ([_|T], N) :- lengthOfList (T, N1),
N is N1 + 1.

Clauses:
lengthOfList (charlist, integer)
lengthOflist (integerlist, integer)

?- lengthOfList ([2, 3, 5], N)

N = 3

?- lengthOfList ([2,3,5], N)
② ↓ N = N1 + 1
lengthOfList ([3,5], N)
② ↓ N = N1 + 1
lengthOfList ([5], N)
② ↓ N = N1 + 1
length of list ([ ], N)
N = N1 + 1 = 0

0 + 1 = 1
1 + 1 = 2
2 + 1 = 3 → N = 3

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 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).

?-
% d:/prolog/problem_04 compiled 0.00 sec, -2 clauses
?- lengthOfList([], N).
N = 0.

?- lengthOfList([1, 10, 100, 1000, 10000, 100000], N).
N = 6.

?- lengthOfList([Deepak, Chourasiya, is , a , future, superstar], N).
N = 6.

?-
```

Problem_04.pl [modified]

File Edit Browse Compile Prolog Pce Help

Problem_04.pl [modified]

```
%Find the length of the list

%length(list, size of list)
lengthOfList([], 0).
lengthOfList([_|T], N):-lengthOfList(T, N1),
                        N is N1+1.
```

Date:

Program 4: Write a prolog program to find the sum of n natural nos. using recursion.

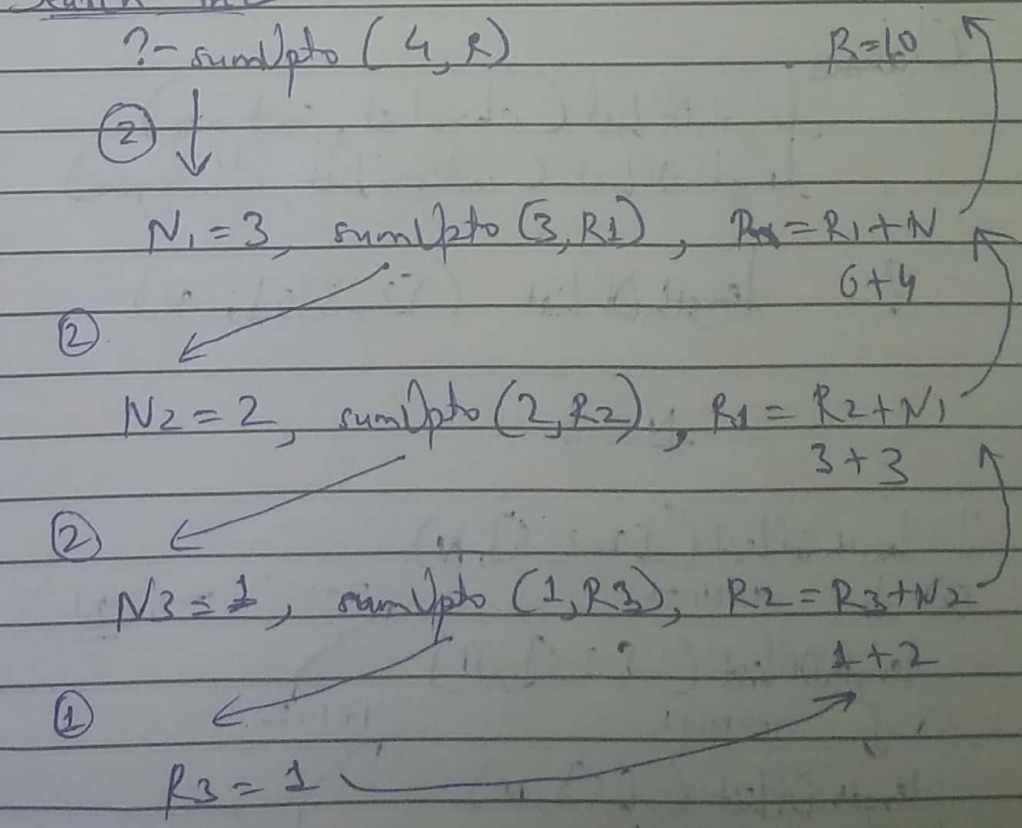
Predicate: sumUpto(integer, integer)

Clauses:

```
sumUpto(1, 1).
sumUpto(N, R) :- N1 = N-1,
                 sumUpto(N1, R1),
                 R = R1+N.
```

```
?- sumUpto(4, R)
R = 10
```

Search Tree:



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 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).

?-
% d:/Prolog/sum_of_natural.pl compiled 0.00 sec, 2 clauses
% d:/prolog/sum_of_natural compiled 0.00 sec, -2 clauses
% d:/prolog/sum_of_natural compiled 0.00 sec, 0 clauses
?- sumUpto(0, Result).
false.

?- sumUpto(1, Result).
Result = 1 ,

?- sumUpto(5, Result).
Result = 15 ,

?- sumUpto(10, Result).
Result = 55 ,

?- sumUpto(20, Result).
Result = 210 ,

?- sumUpto(50, Result).
Result = 1275 ,

?- sumUpto(100, Result).
Result = 5050
```

sum_of_natural.pl [modified]

File Edit Browse Compile Prolog Pce Help

sum_of_natural.pl [modified]

```
%Find the sum of n natural numbers usin recursion
sumUpto(1, 1).
sumUpto(N, R):-N>=2,
              N1 is N-1,
              sumUpto(N1, R1),
              R is R1+N.
```

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 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).

?-
% d:/prolog/fibonacci compiled 0.00 sec, -2 clauses
% d:/prolog/fibonacci compiled 0.00 sec, 0 clauses
?- fib(0, Fibonacci).
Fibonacci = 0 ,

?- fib(1, Fibonacci).
Fibonacci = 1 ,

?- fib(5, Fibonacci).
Fibonacci = 5 ,

?- fib(10, Fibonacci).
Fibonacci = 55 ,

?- fib(30, Fibonacci).
Fibonacci = 832040
```

fibonacci.pl [modified]

File Edit Browse Compile Prolog Pce Help

fibonacci.pl [modified]

```
%Compute fibonacci term using recursion

%Base Cases
fib(0, 0).
fib(1, 1).

%Recursive Cases
fib(N, F):-N1 is N-1,
           N2 is N-2,
           fib(N1, F1),
           fib(N2, F2),
           F is F1+F2.
```

Date:_____

Program 5: Write a prolog program to compute fibonacci term using recursion.

Series (1, 1, 2, 3, 5, 8, 11, 19, ----)

Predicates

fib (integer, integer)

Clauses

```
fib(1,1)
fib(2,1)
fib(N,F) :- N1 = N-1,
            N2 = N1-1,
            fib(N1,F1),
            fib(N2,F2),
            F=F1+F2.
```

? fib(5,F)

F=5

Date:

?- fib(5, F).

↓

N1=4, N2=3, fib(4, F1), fib(3, F2), F=F1+F2

N3=3, N4=2, fib(3, F3), fib(2, F4), fib(3, F2),

F1 = F3+F4.

N5=2, N6=1, fib(2, F5), fib(1, F6), F3=F5+F6,

fib(2, F4), fib(3, F2), F1=F3+F4,

fib(2, F5) fib(1, F6).

F5=1 F6=1 F3=F5+F6

1+1=2

fib(2, F4)

F4=1 F1=F3+F4

=2+1=3

fib(3, F2)

N7=2, N8=1, fib(2, F7), fib(1, F8)

F2=F7+F8, F=F1+F2

F7=1 F8=1 F2=1+1=2, F=3+2

=5

Conclusion: Hence, I have implemented basic prolog program on list, recursion, operators.