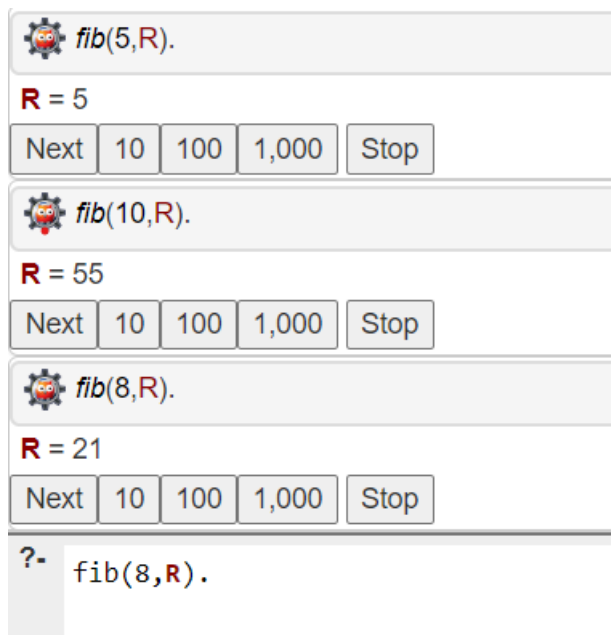


## Practical Assignment For Prolog

1) Write a Prolog predicate `fibonacci/2` to compute the *n*th Fibonacci number.

ANS:

- **Code:**  
fib(0,0).  
fib(1,1).  
fib(N, Result) :-  
    N>0,  
    N1 is N-1,  
    N2 is N-2,  
    fib(N1, R1),  
    fib(N2, R2),  
    Result is R1+R2.
- **Output:**



The screenshot shows a Prolog interpreter interface with three query boxes. Each box contains a query, the result, and a row of buttons: 'Next', '10', '100', '1,000', and 'Stop'.

- Query 1: `fib(5,R).`  
Result: `R = 5`
- Query 2: `fib(10,R).`  
Result: `R = 55`
- Query 3: `fib(8,R).`  
Result: `R = 21`

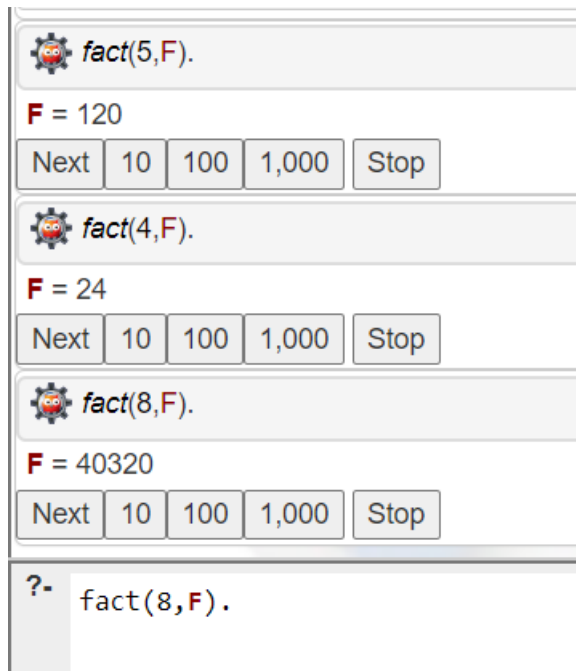
Below the query boxes, there is a prompt `?- fib(8,R).` in a text input field.

2) Write Prolog program to print Factorial

ANS:


- **Code:**  
fact(0, 1).  
fact(N, F) :-  
    N > 0,  
    N1 is N - 1,  
    fact(N1, F1),  
    F is N \* F1.

- **Output:**

 `fact(5,F).`


**F** = 120

Next 10 100 1,000 Stop

 `fact(4,F).`

**F** = 24

Next 10 100 1,000 Stop

 `fact(8,F).`

**F** = 40320

Next 10 100 1,000 Stop

?- `fact(8,F).`


### 3) Calculate distance between two points in 3D Plane

**ANS:**

- **Code:**

distance((X1,Y1,Z1),(X2,Y2,Z2),D):-D is sqrt((X2-X1)^2+(Y2-Y1)^2+(Z2-Z1)^2).

- **Output:**

 `distance((0,0,0),(3,4,5),Result).`

**Result** = 7.0710678118654755

?- `distance((0,0,0),(3,4,5),Result).`

### 4) Check Precedence and associativity of various operators and define new operator.

**ANS:**

- **Code:**

current\_op(P,A,+).

current\_op(P,A,\*).

current\_op(P,A,/).

```
current_op(P,A,-).
current_op(P,A,**).
current_op(P,A,=:).
current_op(P,A,:-).
current_op(P,A,=).
```

- **Code :**

```
op(300, xfx, is_bigger).
elephant is_bigger dog.
```

- **Output:**

```

current_op(P,A,-).
A = fy,
P = 200
A = yfx,
P = 500

current_op(P,A,**).
A = xfx,
P = 200

current_op(P,A,:-).
A = fx,
P = 1200
A = xfx,
P = 1200

current_op(P,A,=:).
A = xfx,
P = 700

current_op(P,A,=).
A = xfx,
P = 700

```

```
% c:/Users/DELL/Documents/Prolog/newopl.pl compiled 0.00 sec, 1 clauses
?- op(300, xfx, is_bigger).
true.

?- elephant is_bigger dog.
true.

?-
```

**5) Write a Prolog program to print out a square of  $n \times n$  given characters on the screen.**

**?- square(5, '\* ').**

\* \* \* \* \*

\* \* \* \* \*

\* \* \* \* \*

\* \* \* \* \*

\* \* \* \* \*

**ANS:**

- **Code:**

```
square(N, Char) :-  
    length(Row, N),  
    maplist(=(Char), Row),  
    length(Square, N),  
    maplist(=(Row), Square),  
    maplist(writeln, Square).
```

- **Output:**

```
⚙️ square(5, '*').  
[*, *, *, *, *]  
[*, *, *, *, *]  
[*, *, *, *, *]  
[*, *, *, *, *]  
[*, *, *, *, *]  
true  
⚙️ square(6, '*').  
[*, *, *, *, *, *]  
[*, *, *, *, *, *]  
[*, *, *, *, *, *]  
[*, *, *, *, *, *]  
[*, *, *, *, *, *]  
[*, *, *, *, *, *]  
true  
?- square(6, '*').
```