

**Roll Number: 9237**

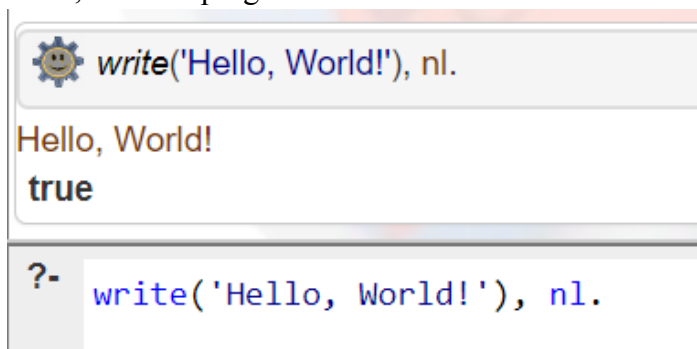
**Name: Valiaparambil Ryan Taffy**

## **Experiment No: 6**

**Title:** Prolog Programming Set 1

**Objective:** To get acquainted with logical programming  
**Implement**

1. Hello, World!" program



The image shows a Prolog environment window. At the top, there is a gear icon followed by the code `write('Hello, World!'), nl.`. Below this, the output `Hello, World!` is displayed in orange text, followed by `true` in black text. At the bottom, there is a prompt `?-` followed by the code `write('Hello, World!'), nl.`.

2. Program to check if an element is a member of a list



The image shows a Prolog environment window. At the top, there is a tab labeled "Program" with a close button and a plus sign. Below the tab, there is a list of two Prolog rules:  
1 `member(X, [X|_]).`  
2 `member(X, [_|T]) :- member(X, T).`  
Below the rules, there is a gear icon followed by the code `member(5, [1,2,4,5]).`. Below this, the output `true` is displayed. Below the output, there are four buttons: "Next", "10", "100", and "1,000", followed by a "Stop" button. Below the buttons, there is a gear icon followed by the code `member(3, [1,2,5,4]).`. Below this, the output `false` is displayed in red text.

3. Program to append two lists

 `A = [1, 5], B=[12, 14, 19, 30], append(A, B, Y)`





`A = [1, 5],`

`B = [12, 14, 19, 30],`

`Y = [1, 5, 12, 14, 19, 30]`


?- `A = [1, 5], B=[12, 14, 19, 30], append(A, B, Y)`

4. Program to reverse a list

  Program  

1 `reverse([], []).`

2 `reverse([H|T], R) :- reverse(T, TR), append(TR, [H], R).`

 `reverse([10, 9, 8, 7], R).`

`R = [7, 8, 9, 10]`





5. Program to find the length of a list

 `length([1,2,3,4],N).`

`N = 4`


?- `length([1,2,3,4],N).`

6. Program to find the maximum of two numbers

  Program  

1 `max(X, Y, X) :- X >= Y.`



2 `max(X, Y, Y) :- Y > X.`

 `max(48, 82, Max).`


**Max** = 82

?- `max(48, 82, Max).`

7. Program to find the factorial of a number

  Program × +



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

 `factorial(5, F).`


**F** = 120

Next 10 100 1,000 Stop

8. Program to find the nth Fibonacci number

  Program × +

```
1 fibonacci(0, 0).
2 fibonacci(1, 1).
3 fibonacci(N, F) :- N > 1, N1 is N-1, N2 is N-2, fibonacci(N1, F1), fibonacci(N2, F2),
```





 `fibonacci(6, F).`

**F** = 8


Next 10 100 1,000 Stop

?- `fibonacci(6, F).`

9. Program to find the sum of a list of numbers

  Program  



```
1 sum([], 0).  
2 sum([H|T], S) :- sum(T, S1), S is S1+H.
```

 `sum([1, 2, 3, 4, 5], S).`


**S** = 15

?- `sum([1, 2, 3, 4, 5], S).`

10. Program to find the smallest element in a list.

  Program  

```
1 min_list([H|T], Min) :- min_list(T, H, Min).  
2 min_list([], Min, Min).  
3 min_list([H|T], MinSoFar, Min) :- H < MinSoFar, min_list(T, H, Min).  
4 min_list([H|T], MinSoFar, Min) :- H >= MinSoFar, min_list(T, MinSoFar, Min).  
5 |
```

 `min_list([5, 4, 3, 2, 1], Min).`

**Min** = 1

Next 10 100 1,000 Stop