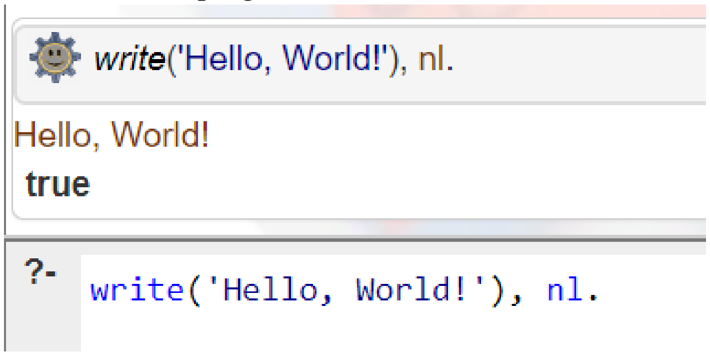
Roll Number: 9237
Name: Valiaparambil Ryan Taffy

Experiment No: 6

Title: Prolog Programming Set 1

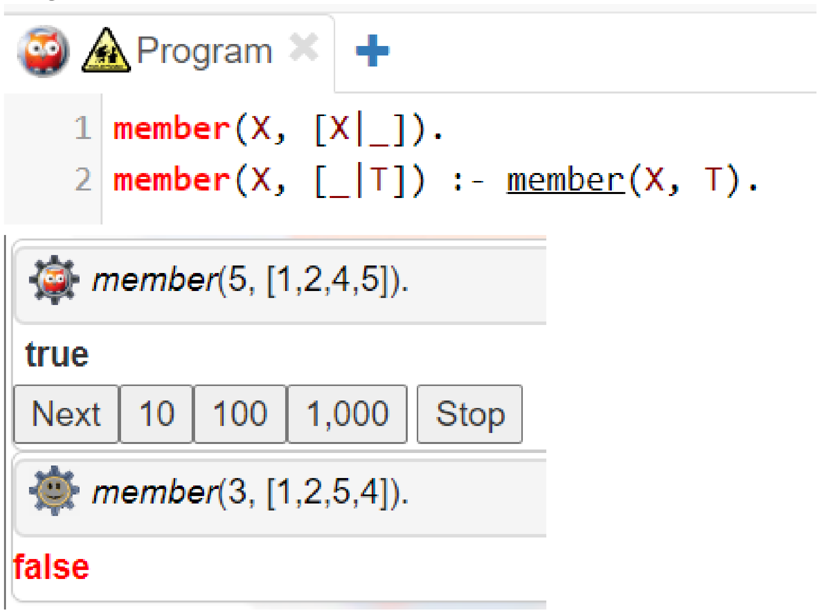
Objective: To get acquainted with logical programming
Implement

1. Hello, World!" program



```
write('Hello, World!'), nl.
Hello, World!
true
?- write('Hello, World!'), nl.
```

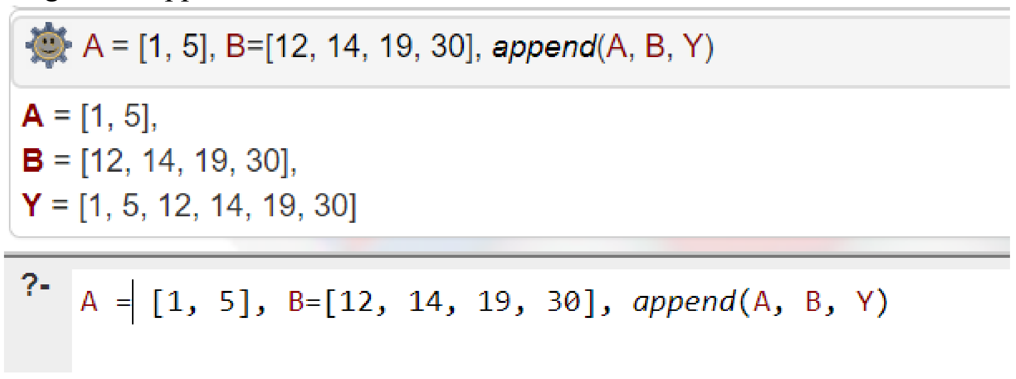
2. Program to check if an element is a member of a list



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

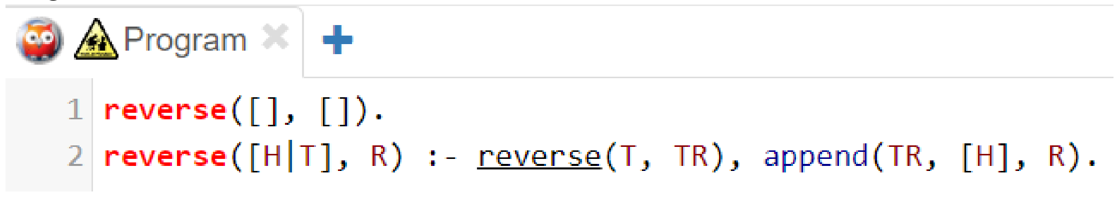
```
member(5, [1,2,4,5]).
true
member(3, [1,2,5,4]).
false
```

3. Program to append two lists

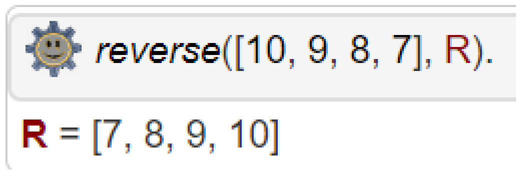


```
A = [1, 5], B=[12, 14, 19, 30], append(A, B, Y)

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

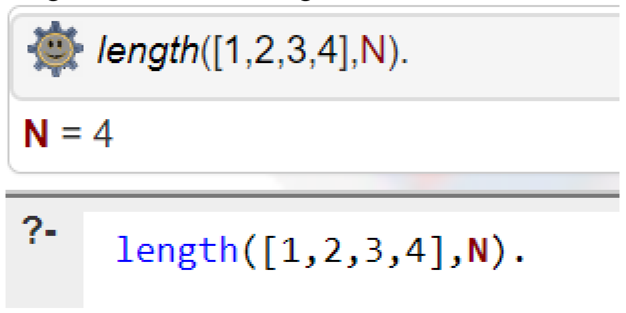


```
reverse([], []).
reverse([H|T], R) :- reverse(T, TR), append(TR, [H], R).
```



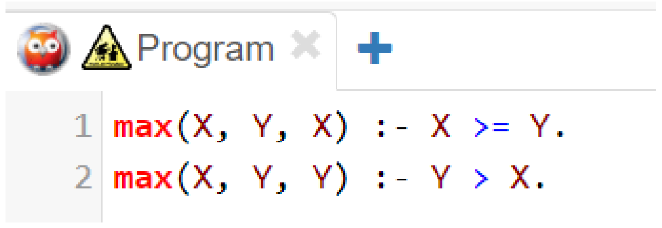
```
reverse([10, 9, 8, 7], R).

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

5. Program to find the length of a list

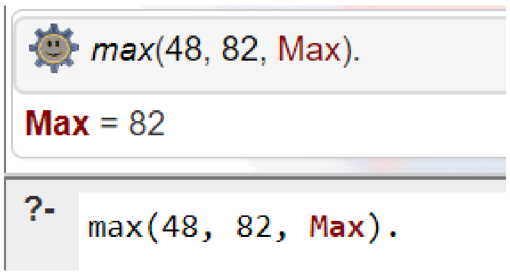


```
length([1,2,3,4],N).

N = 4

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

6. Program to find the maximum of two numbers

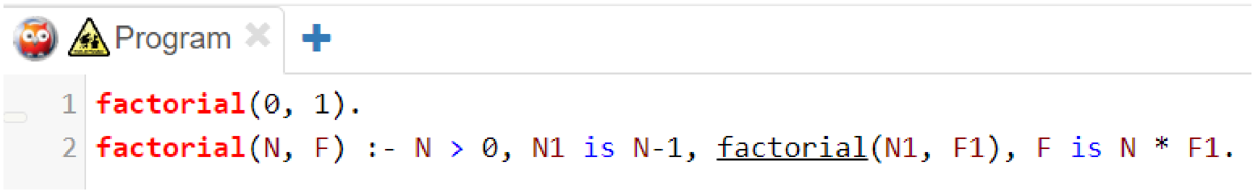


```
max(X, Y, X) :- X >= Y.
max(X, Y, Y) :- Y > X.
```

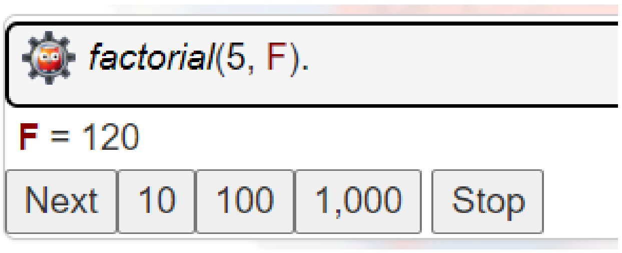


```
max(48, 82, Max).
Max = 82
?- max(48, 82, Max).
```

7. Program to find the factorial of a number

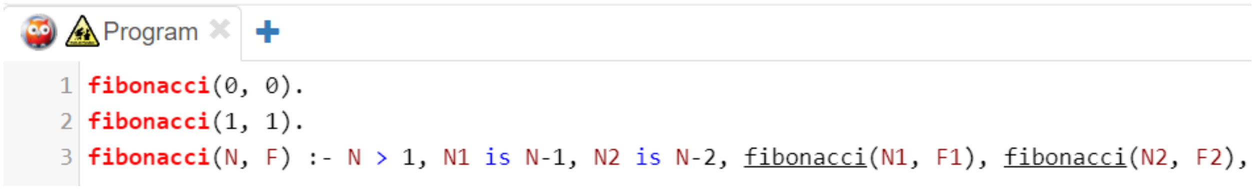


```
factorial(0, 1).
factorial(N, F) :- N > 0, N1 is N-1, factorial(N1, F1), F is N * F1.
```

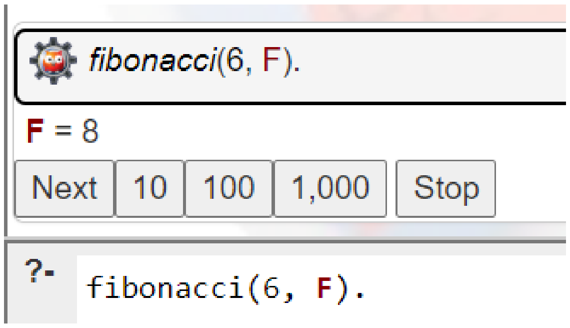


```
factorial(5, F).
F = 120
```

8. Program to find the nth Fibonacci number



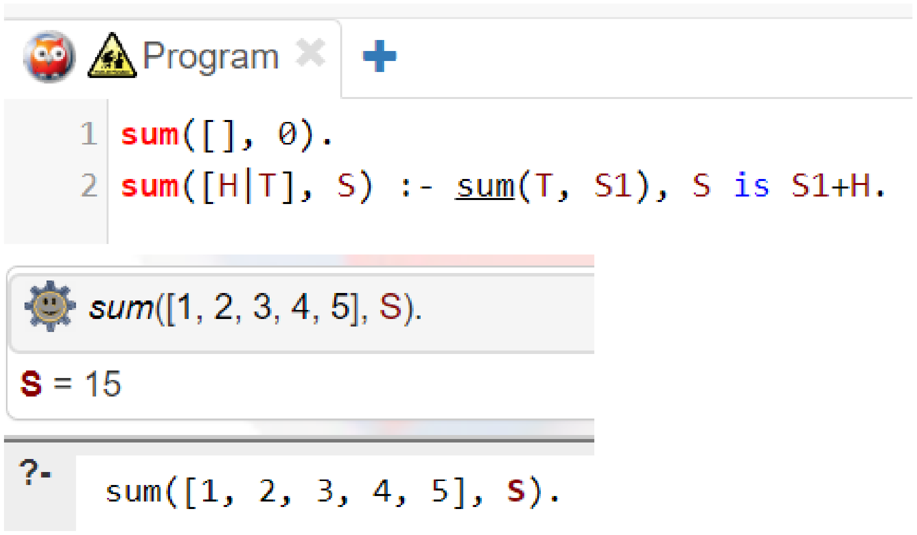
```
fibonacci(0, 0).
fibonacci(1, 1).
fibonacci(N, F) :- N > 1, N1 is N-1, N2 is N-2, fibonacci(N1, F1), fibonacci(N2, F2),
```



```
fibonacci(6, F).
F = 8
```

```
?- fibonacci(6, F).
```

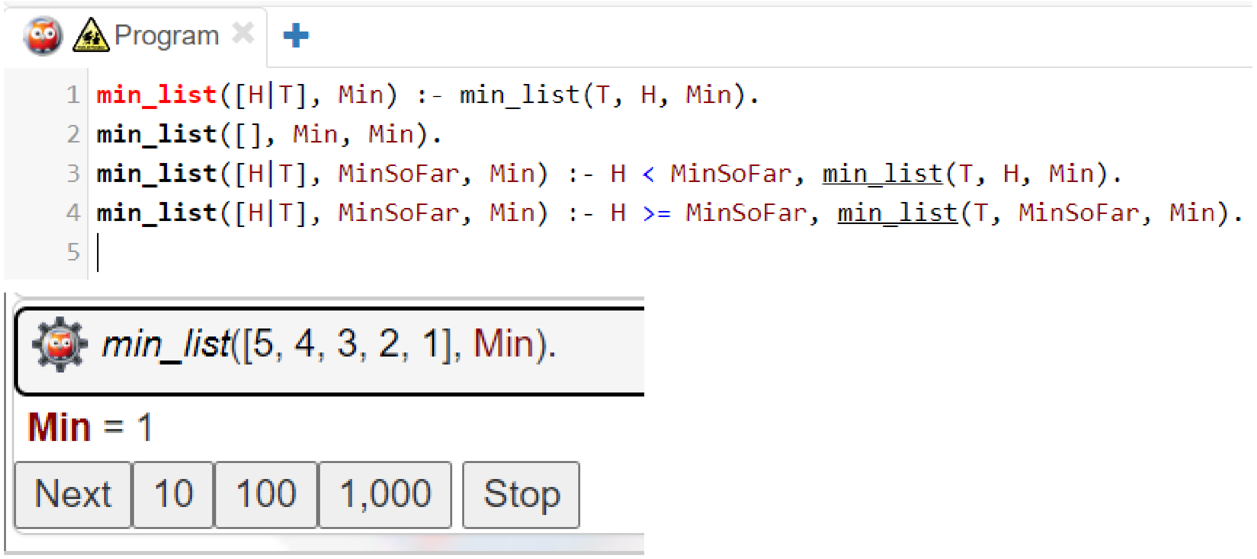
9. Program to find the sum of a list of numbers



```
sum([], 0).
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.



```
min_list([H|T], Min) :- min_list(T, H, Min).
min_list([], Min, Min).
min_list([H|T], MinSoFar, Min) :- H < MinSoFar, min_list(T, H, Min).
min_list([H|T], MinSoFar, Min) :- H >= MinSoFar, min_list(T, MinSoFar, Min).

```

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