

**Department of Computer Engineering**

**Academic Term II : 22-23**

**Class: B.E (Comp A), Sem VI**

**Subject Name: Artificial Intelligence**

**Student Name: Gouri Sankhe**

**Roll No: 9230**

<b>Practical No:</b>	<b>6</b>
<b>Title:</b>	<b>Prolog Programming Set 1</b>
<b>Date of Performance:</b>	<b>20 -03- 2023</b>
<b>Date of Submission:</b>	<b>3 -04- 2023</b>

**Rubrics for Evaluation:**

<b>Sr. N o</b>	<b>Performance Indicator</b>	<b>Excellent</b>	<b>Good</b>	<b>Below Average</b>	<b>Marks</b>
1	On time Completion & Submission (01)	01 (On Time )	NA	00 (Not on Time)	
2	Logic/Algorithm Complexity analysis(03)	03(Correct)	02(Partial)	01 (Tried)	
3	Coding Standards (03): Comments/indentation/Naming conventions Test Cases /Output	03(All used)	02 (Partial)	01 (rarely followed)	
4	Post Lab Assignment (03)	03(done well)	2 (Partially Correct)	1(submitted)	
<b>Total</b>					

**Signature of the Teacher** :

Implementation:

### 1. Hello, World! program

<pre>1 :- initialization(main). 2 main :- write('Hello World!').</pre>	<pre>GNU Prolog 1.5.0 (64 bits) Compiled Feb 22 2023, 13:01:45 with gcc Copyright (C) 1999-2023 Daniel Diaz  compiling /home/cg/root/6432d47474a97/main.pg for byte code... /home/cg/root/6432d47474a97/main.pg compiled, 1 lines read - 311 bytes written, 3 ms Hello World!! ?-</pre>
--	---

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

<pre>member(X, [_ _]). member(X, [_ T]) :- member(X, T).</pre>	<pre>\$ cd /path/to/program/directory \$ swipl Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.4)  ?- [membership]. true.  ?- member(3, [1, 2, 3, 4, 5]). true.  ?- member(6, [1, 2, 3, 4, 5]). false.  ?- halt.</pre>
--	--

### 3. Program to append two lists

<pre>append([], L, L). append([H T], L, [H R]) :- append(T, L, R).</pre>	<pre>\$ cd /path/to/program/directory \$ swipl Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.4)  ?- [append_lists]. true.  ?- append([1, 2, 3], [4, 5, 6], Result). Result = [1, 2, 3, 4, 5, 6].  ?- append([], [1, 2, 3], Result). Result = [1, 2, 3].  ?- halt.</pre>
--	---

### 4. Program to reverse a list

<pre>reverse([], []). reverse([H T], R) :- reverse(T, TR), append(TR, [H], R).</pre>	<pre>\$ cd /path/to/program/directory \$ swipl Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.4)  ?- [reverse_list]. true.  ?- reverse([1, 2, 3, 4, 5], Result). Result = [5, 4, 3, 2, 1].  ?- reverse([], Result). Result = [].  ?- halt.</pre>
--	---

### 5. Program to find the length of a list

```
length([], 0).
length([_:T], Len) :- length(T, Len1), Len is Len1 + 1.
```

```
$ cd /path/to/program/directory
$ swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.4)

?- [list_length].
true.

?- length([1, 2, 3, 4, 5], Len).
Len = 5.

?- length([], Len).
Len = 0.

?- halt.
```

## 6. Program to find the maximum of two numbers

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

```
$ cd /path/to/program/directory
$ swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.4)

?- [max_two_numbers].
true.

?- max(10, 20, Max).
Max = 20.

?- max(100, 50, Max).
Max = 100.

?- max(5, 5, Max).
Max = 5.

?- halt.
```

## 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.
```

```
$ cd /path/to/program/directory
$ swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.4)

?- [factorial].
true.

?- factorial(5, F).
F = 120.

?- factorial(0, F).
F = 1.

?- halt.
```

## 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), F is F1 + F2.
```

```
$ cd /path/to/program/directory
$ swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.4)

?- [fibonacci].
true.

?- fibonacci(6, F).
F = 8.

?- fibonacci(10, F).
F = 55.

?- halt.
```

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

```
sum_list([], 0).
sum_list([H|T], Sum) :- sum_list(T, Sum1), Sum is H + Sum1.
```

```
$ cd /path/to/program/directory
$ swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.4)

?- [sum_list].
true.

?- sum_list([1, 2, 3, 4, 5], Sum).
Sum = 15.

?- sum_list([10, 20, 30, 40], Sum).
Sum = 100.

?- halt.
```

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

```
smallest([X], X).
smallest([H|T], X) :- smallest(T, X1), (H < X1 -> X = H ; X = X1).
```

```
$ cd /path/to/program/directory
$ swipl
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.4)

?- [smallest].
true.

?- smallest([5, 2, 9, 3, 7], X).
X = 2.

?- smallest([100, 200, 50, 300, 150], X).
X = 50.

?- halt.
```