Lab 3: Recursion (Prolog /C/Java/Python)

Objective:

The objective of this lab is to understand and implement recursion in Prolog or any other high level language by writing two programs:

1. A recursive program to compute the factorial of a number, where the number is read from the user.
2. A recursive solution to the Tower of Hanoi (TOH) problem.

Theory

Recursion: ……..

……

1. **Factorial:**
   - The factorial of a number `N`, denoted as `N!`, is the product of all positive integers up to `N`. The recursive definition is:
     - `0! = 1` (base case)
     - `N! = N * (N-1)!` for `N > 0` (recursive case)
2. **Tower of Hanoi (TOH):**
   - The TOH problem involves moving a stack of disks from one peg to another, using an auxiliary peg, under the constraints that only one disk can be moved at a time, and no larger disk can be placed on a smaller disk. The recursive solution involves:
     - Moving `N-1` disks from the source peg to the auxiliary peg.
     - Moving the largest disk directly to the destination peg.
     - Moving the `N-1` disks from the auxiliary peg to the destination peg.

Program 1: Recursive program to find the Factorial of a number

```
% Factorial base case: The factorial of 0 is 1

factorial(0, 1).

% Recursive case: The factorial of N is N * factorial of (N-1)

factorial(N, Result) :-

  N > 0,

  N1 is N - 1,

  factorial(N1, R1),

  Result is N * R1.
```

```
% Read a number from the user and compute its factorial
factorial_user :-
   write('Enter a number: '),
   read(N),
   factorial(N, Result),
   format('The factorial of ~w is ~w.', [N, Result]).
```

Program 2: Tower of Hanoi (TOH) Problem

```
% TOH base case: Moving 1 disk from source to destination
move(1, Source, Destination, _) :-
   format('Move disk 1 from ~w to ~w.~n', [Source, Destination]).
% Recursive case: Moving N disks from source to destination
move(N, Source, Destination, Auxiliary) :-
   N > 1,
   N1 is N - 1,
   move(N1, Source, Auxiliary, Destination),
   format('Move disk ~w from ~w to ~w.~n', [N, Source, Destination]),
   move(N1, Auxiliary, Destination, Source).
% Tower of Hanoi solution for N disks
tower_of_hanoi(N) :-
   move(N, 'Source', 'Destination', 'Auxiliary').
```

Conclusion: