**Experiment No:** 5

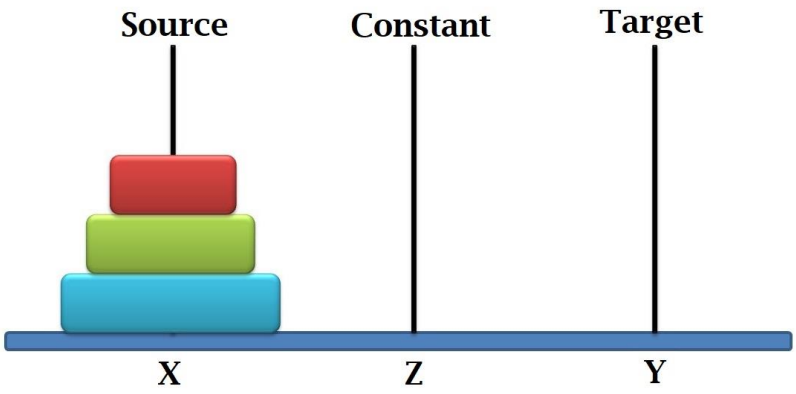
**Experiment Name:** Write a Program in PROLOG to Solve a Tower of Hanoi Problem.

## Towers of Hanoi Problem

Towers of Hanoi Problem is a famous puzzle to move N disks from the source peg/tower to the target peg/tower using the intermediate peg as an auxiliary holding peg. There are two conditions that are to be followed while solving this problem −

- A larger disk cannot be placed on a smaller disk.
- Only one disk can be moved at a time.

The following diagram depicts the starting setup for N=3 disks.




To solve this, we have to write one procedure move **(N, source, target, constant).** Here N number of disks will have to be shifted from Source peg to Target peg keeping Constant peg as intermediate.

For example – move(3, source, target, constant).

Codes:

```
move(1,X,Y,_):-
   write('move top disk from '),
   write(X),
   write(' to '),
   write(Y),
   nl.

move(N,X,Y,Z):-
   M is N-1,
   move(M,X,Z,Y),
   move(1,X,Y,_),
   move(M,Z,Y,X).
```

Let's go through the code step by step:

- `move(1,X,Y,_):-`
  - This is the base case for the recursive `move/4` predicate. It says that when there is only one disk to move (i.e., `N` is 1), you should execute the following instructions:
    1. `write('move top disk from '),`: This line prints the message "move top disk from " to the console.
    2. `write(X),`: This line prints the source peg (`X`) to the console.
    3. `write(' to '),`: This line prints the message " to " to the console.
    4. `write(Y),`: This line prints the destination peg (`Y`) to the console.
    5. `nl.`: This line adds a newline character to the console output, making it more readable.

- `move(N, X, Y, Z):-`
  - This is the beginning of a Prolog rule definition for the `move/3` predicate. It states that to move N disks from peg X to peg Y using peg Z as an auxiliary peg, you can perform the following steps.
- `M is N-1,`
  - This line calculates M as one less than N. In the context of the Towers of Hanoi problem, this is used to represent the smaller subproblem of moving N-1 disks.
- `move(M, X, Z, Y),`
  - This recursive call is used to move the top M (N-1) disks from peg X to peg Z using peg Y as an auxiliary peg. This step recursively solves the subproblem of moving the smaller disks to the auxiliary peg.

- `move(1, X, Y, _),`
  - This line is used to move the largest disk, which is the disk at the bottom, from peg X to peg Y. The `_` is a placeholder for an auxiliary peg, indicating that it doesn't matter which auxiliary peg is used in this step.
- `move(M, Z, Y, X).`
  - This is the final recursive call and is used to move the M (N-1) disks from peg Z to peg Y using peg X as an auxiliary peg. This step is the final move to place the smaller disks on the destination peg.

The code continues to recursively call `move/3` with smaller subproblems until it reaches the base case, where only one disk needs to be moved. Then, it starts moving the disks back from the auxiliary peg to the destination peg while maintaining the order of disk sizes.

When you invoke `move/3` with appropriate values for N, X, Y, and Z, Prolog will execute these rules to generate a sequence of moves that solve the Towers of Hanoi puzzle with N disks, moving them from peg X to peg Y using peg Z as an auxiliary peg.