**Towers of Hanoi**

The Tower of Hanoi (also called the Tower of Brahma or Lucas' Tower[1] and sometimes pluralized) is a mathematical game or puzzle. It consists of three rods and a number of disks of different sizes, which can slide onto any rod. The puzzle starts with the disks in a neat stack in ascending order of size on one rod, the smallest at the top, thus making a conical shape.

***The objective of the puzzle is to move the entire stack to another rod, obeying the following simple rules:***

**Rule 1:** Only one disk can be moved at a time.

**Rule 2:** Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack or on an empty rod.

**Rule 3:** No disk may be placed on top of a smaller disk.

***Algorithm***

***Move Disks***

|  |  |
| --- | --- |
| **Step 1** | Start Process |
| **Step 2** | Read number of disks |
| **Step 3** | Call Move Disks with number of disks, source, destination, aux |
| **Step 4** | Stop Process |

***Move Disks***

|  |  |
| --- | --- |
| **Step 1** | Start Process |
| **Step 2** | Receive number of disks and store in n |
| **Step 3** | If n is equal to 1 then Move disk from source to destination and goto Step 16 |
| **Step 4** | Else goto Step 5 |
| **Step 5** | Assign disks as n-1 |
| **Step 6** | Assign source as source |
| **Step 7** | Assign destination as aux |
| **Step 8** | Assign aux as destination |
| **Step 9** | Call Move Disks with disks, source, destination, aux |
| **Step 10** | Move disk n from source to destination |
| **Step 11** | Assign disks as n – 1 |
| **Step 12** | Assign source as aux |
| **Step 13** | Assign destination as destination |
| **Step 14** | Assign aux as source |
| **Step 15** | Call Move Disks with disks, source, destination, aux |
| **Step 16** | Stop Process |

***Flow Chart***

***Pseudo Code***

***Main Pseudo Code***

START

READ number of disks

CALL move\_disks(no of disks, source, destination, aux)

STOP

***Move Disks Sub Routine PseudoCode***

START

PROCEDURE: move\_disks(disks, source, destination, aux)

IF disks equal to 1

Move Disks from source to destination

ELSE

CALL move\_disks(disks - 1, source, aux, destination)

Move Disks from source to destination

CALL move\_disks(disks - 1, aux, destination, source)

END IF

END PROCEDURE

STOP