Tower of Hanoi

"Not every puzzle is intended to be solved. Some are in place to test your limits. Others are, in fact, not puzzles at all"

The tower of Hanoi (also called Tower of Brahma or Lucas' Tower) is a mathematical game or puzzle invented by French mathematician **Edouard Lucas** in 1883.

Here is the puzzle:

```
Question:
--------------
1. There are 3 towers. Tower 1 has n disks, where n is a positive number.
   Tower 2 and 3 are empty.

2. The disks  are increasingly placed in terms of size such that
   the smallest disk is on top and largest disk is at the bottom.

3. You are required to
    3.1 Print the instructions to move the disks
    3.2 From Tower 1 to Tower 2 using Tower 3 and
    3.3 Following the rules
        3.3.1 Move 1 disk at a time
        3.3.2 Never place a smaller disk under a larger disk
        3.3.3 You can only move a disk at the top
4.Write a recursive logic.
```

And the required code to be completed:

```
import java.io.*;
import java.util.*;
public class Main {
    public static void main(String args) throw Exception {
        //Write your code here

    }

    public static void toh(int n,int t1id, int t2id, int t3id){
        //complete the given function
    }
}
```

The function " **toh()**" is to be designed as a recursive function which will solve the tower of Hanoi puzzle.

For any recursive algorithmic design, there are two parts of thinking that come into play which are:

1. **HIGH LEVEL THINKING**
2. **LOW LEVEL THINKING**

In High Level Thinking,

Step 1: We establish our ***expectation*** from the function toh(), which is that it will know the valid set of instructions that will help us move the required number of ***n*** disks from tower 1 to tower 2.

Step 2: We have ***faith*** toh(n-1,..) knows the valid set of instructions to move ***n -1*** disks from tower 1 to tower 2.

Step 3: We establish a relation/link between the expectation and faith (through a recursive call).

This is just like a proof problem of PMI where we assume the given equation to be true for **n = k** and then prove it for **n = k +1** , which is basically assuming that the smaller part of the bigger problem is true, and then, acting upon the bigger fragment of the problem accordingly.

In Low level thinking, we choose the suitable base case for termination of recursion.

The code implementation for solving the tower of Hanoi is as follows:

```java
import java.io.*;
import java.util.*;
public class Main {
    public static void main(String args) throw Exception {
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        int t1 = scn.nextInt();
        int t2 = scn.nextInt();
        int t3 = scn.nextInt();

        toh(n,t1,t2,t3);

    }

    public static void toh(int n,int t1id, int t2id, int t3id){
        if(n == 0)
        {
            return;
        }
        toh(n-1,t1id,t3id,t2id);// will print the instructions to move n -1 disks from t1 to t3 using t2
        System.out.println(n + "[" t1id + "->" + t2id + "]");
        toh(n-1,t3id,t2id,t1id);//will print the instructions to move n - 1 disks from t3 to t2 using t1
    }
}
```

Sample Input:

3

10

11

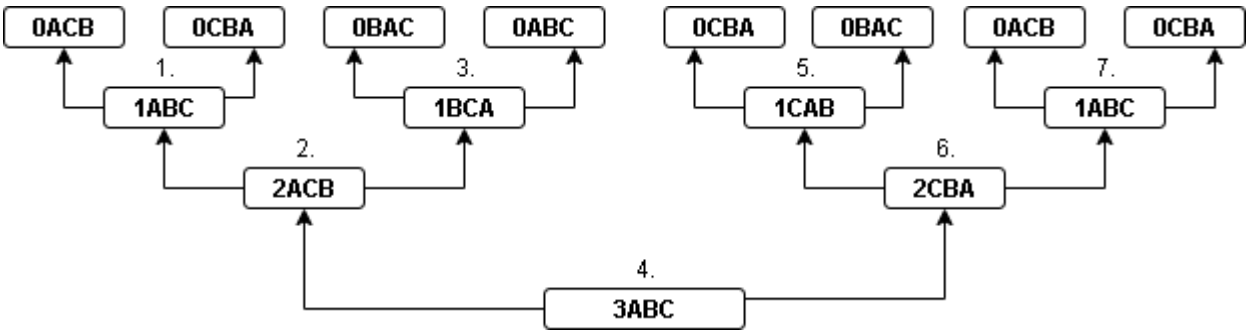
12

Sample Output:

```
1[10->11]
2[10->12]
1[11->12]
3[10->11]
1[12->10]
2[12->11]
1[10->11]
```

The logic that goes behind this is simple, each **toh()** makes two calls. We can call them left call and right call.

Taking the example of 3 disks with tower names as A, B, C:



Left Call **Right Call**

This is a recursion tree where the calls made in the entire procedure have been shown.

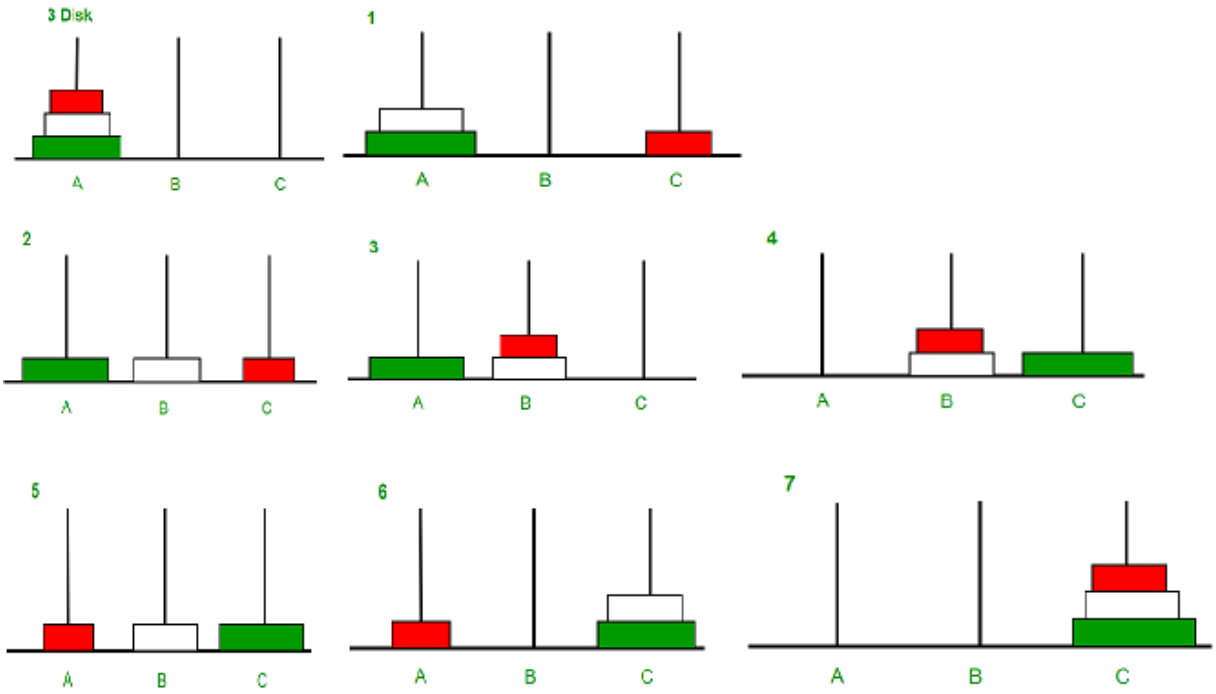
We find the valid inorders in the trees and operate accordingly.

The orders in which the shift transfers take place have been written in serial order:

1AB-> 2AC-> 1BC-> 3AB-> 1CA-> 2CB-> 1AB

Where the numeric value describes the disk number and the following alphabets denote the source and destination towers respectively in a transfer.

Step by Step illustration of the Tower of Hanoi problem:



In this illustration the starting tower is **A** and the destination tower is **C.**

The Tower of Hanoi is a classical problem of recursion.

The spirit to solving this or any recursion problem is having belief in the faith you have selected for the smaller problem.

"*Only suspension of disbelief will help you cross the vast ocean that is called recursion"*