

DAY  
44  
DSA

20 Oct, 2021

## Recursion Continued .....

Agenda for today

1. Print Zigzag → CW
2. Tower of Hanoi → CW.

Ques. 1.

## PRINT Zigzag

We are given an integer  $n$  and required to print a particular type of sequence of numbers  
function name  $\rightarrow P_{Zz}$

Eg-1.

Input  $\rightarrow 1$   
Output  $\rightarrow \underline{1} \underline{1} \underline{1}$

$P_{Zz}(0) \quad P_{Zz}(0)$

Eg-2

Input  $\rightarrow 2$   
Output  $\rightarrow \underline{2} \underline{1} \underline{1} \underline{1} \underline{2} \underline{1} \underline{1} \underline{1} \underline{2}$

$P_{Zz}(1) \quad P_{Zz}(1)$

Eg-3

Input  $\rightarrow 3$   
Output  $\rightarrow \underline{3} \underline{2} \underline{1} \underline{1} \underline{2} \underline{1} \underline{2} \underline{3} \underline{2} \underline{1} \underline{1} \underline{2} \underline{1} \underline{2} \underline{3}$

Input  $\rightarrow 2$

Output  $\rightarrow 2 \underline{1} \underline{1} \underline{1} \underline{2} \underline{1} \underline{1} \underline{1} \underline{2}$

} We need to build  
faith that we  
will receive this  
Output for  $n-1$

Input  $\rightarrow 3$

Output  $\rightarrow \underline{3} \underline{2} \underline{1} \underline{1} \underline{1} \underline{2} \underline{1} \underline{1} \underline{2} \underline{3}$

Output for  $n=2$

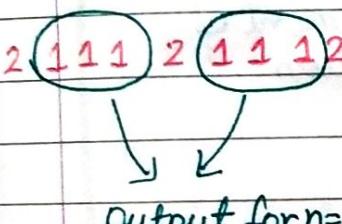
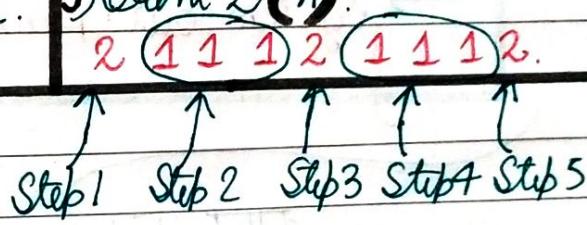
Output for  $n=2$

## Analyzing the output for $n=3$ .

1. Printed 3.
2. Printed the output for  $n=2$
3. Printed 3.
4. Printed the output for  $n=2$ .
5. Printed 3.

Page No.	
Date	

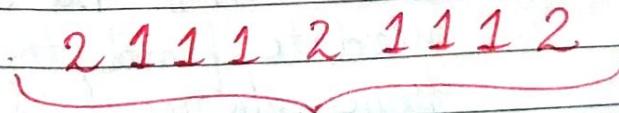
### High Level Thinking for $n=2$ .

Expectation	Faith	Expectation meets/ needs faith
for $n=2$ print 3   Output for $n=1$	for $n=1$ . print <b>1 1 1</b>  We have a strong faith that our function knows how to print for $n=1$ i.e. $n-1$ .	for $n=2$ .  1) Print 2( $n$ ). 2) Let function pzz.print output for $n=1$ ( $n-1$ ) 3) Print 2( $n$ ). 4) Let function pzz.print the output for $n=1$ ( $n-1$ ). 5) Print 2( $n$ ).   Step 1 Step 2 Step 3 Step 4 Step 5

### High Level Thinking

#### 1. → Expectation

→ We expect that our code would the  
print the correct output for  $n$ .  
if  $n=2$

Output → 

We expect that the correct  
output will be printed

#### 2. → Faith

→ We form a faith that our code will be  
able to print the output for  $n-1$ .  
→ If you fit  $n-1$  fit output  
→ If we assume it's correct faith right.  
→ If  $n-1$  fit correct output then /.

if  $n=1$

Output → 

We assume that we will receive this output

### 3. Expectation meets Faith

Page No.	
Date	

We have 5 steps here.

- Point n
- Let function print for  $(n-1)$  output
- Point n
- Let function print output for  $(n-1)$
- Point n return type void नहीं हो सकता तो वार्ता नहीं कर सकता फूल  
function name is pzz

public static void pzz(int n)

- 1 → System.out.print(n + " ");
- 2 → pzz(n-1);
- 3 → System.out.print(n + " ");
- 4 → pzz(n-1);
- 5 → System.out.print(n + " ");

$n$  is passed as the argument of function

माने High Level Thinking से ये code नहीं लिखा गया

but this code is not complete.

because code की अभी ज़रूरी पृष्ठी नहीं है।

जो उसे लिखना चाहिए। So, basically we

need to find the Base Case & How do we find the Base Case?

We find the Base Case by doing the Low Level Thinking.



Dry run & decouple the stack frame to find the Base Case.

## Low Level Thinking

→ इस ques में एक func है 2

Calls हो रही हैं.

Page No. \_\_\_\_\_

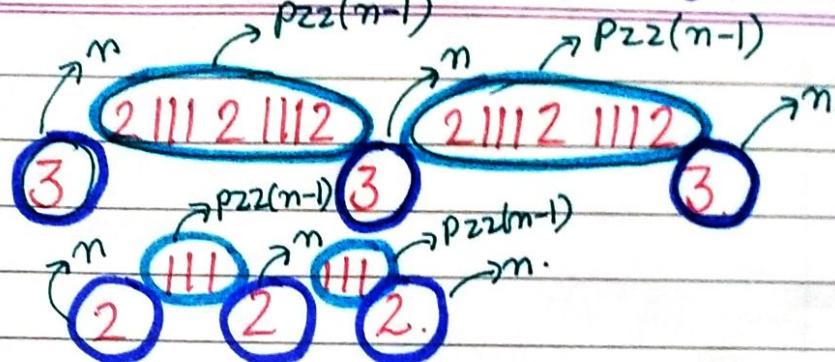
Date: \_\_\_\_\_

Eg →  $n=3$

2 calls समाजन के लिए ये ques हैं।

Expectation :-

Faith :-



Expectation meets faith

→  $P_{22}(3)$  को express करना है  $P_{22}(2)$  की terms में।

System.out.print( $n + " "$ ); → Line a  
 $P_{22}(n-1)$ ; → Line b  
System.out.print( $n + " "$ ); → Line c.  
 $P_{22}(n-1)$ ; → Line d.  
System.out.print( $n + " "$ ); → Line e.

→ Line b 3 टोड़ा  
फल Line d पर  
function को call  
line महसूस 2.  
recursive calls  
1st function में

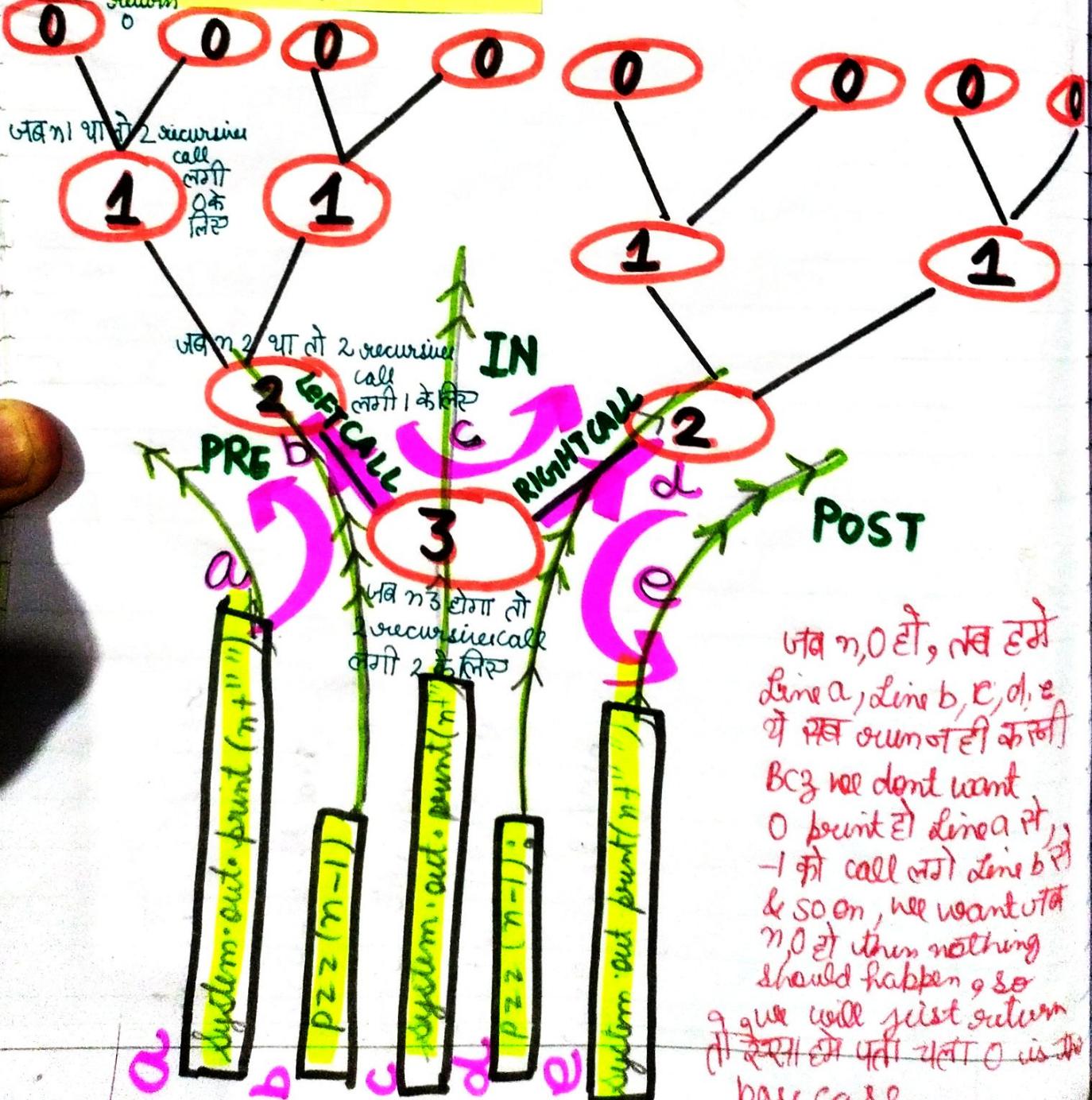
```
public static void  
pzz(uint n) {
```

System.out.print("n+ ")); → Line a → **Prearea**  
 pzz(n-1); → Line b **Left Call** → **Fnarea**  
 System.out.print("n+ ")); → Line c  
 pzz(n-1); → Line d **Right Call** → These 2 are recursive calls तो वही फ़र्माए  
 System.out.print("n+ ")); → Line e  
 Postarea → functions में 2 recursive call हो सकते हैं तो समझें।

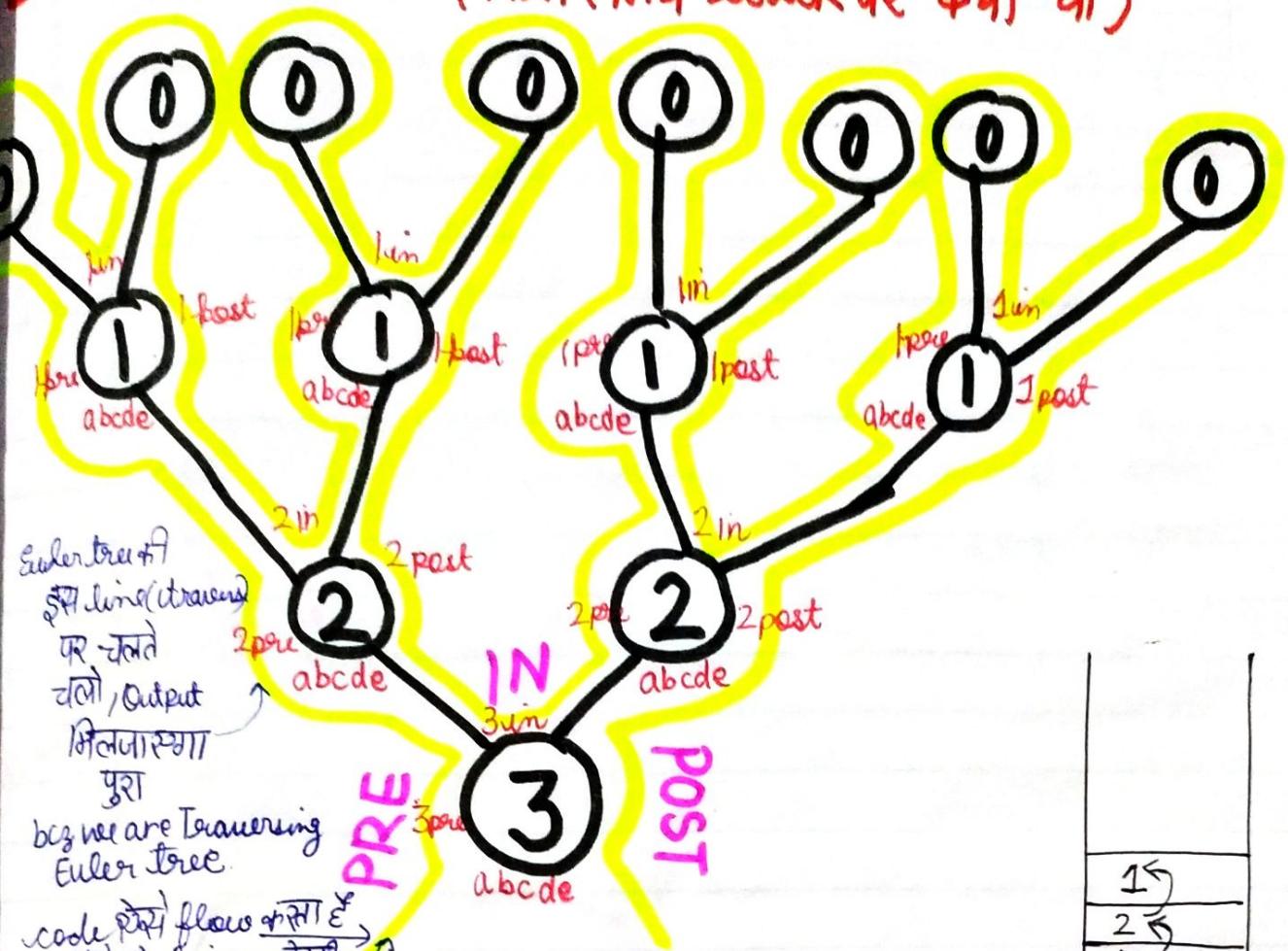
3. ज्वरों के नियन्त्रण के लिए अलग-अलग इलेक्ट्रोडों का उपयोग किया जाता है।

- Line a → Bee area
- Line b → Left Call
- Line c → In area
- Line d → Right call
- Line e → Post area

These 2 are recursive  
 calls तो परीक्षा  
 function में 2 recursive  
 call किसायाएं तो समझ  
 जाओ कि Euler tree  
 बनाके देखना है।  
 function में 2 या 2 से ज्यादा  
 recursive call मिलते  
 Euler tree बनाना है।



Euler tree में stack की timeline कैसे हैं  
(किस समय stack पर क्या था)



```
public static void pzz(uint n)
```

```
{ if(n==0) } → Base Case
```

```
    { return; }
```

```
    System.out.println(n+"pre");
```

```
    pzz(n-1);
```

```
    System.out.println(n+"in");
```

```
    pzz(n-1);
```

```
    System.out.println(n+"post");
```

```
} Output →
```

```
3pre
2pre
1pre
1in
1post
2in
1pre
1in
1post
2post
3in
2pre
1pre
1in
1post
2in
1pre
1in
1post
2post
3post
```

जैसी ही 5 line होती है  
pzz(n) memory से वापिस  
out हो जाता है उसे the  
help of Garbage  
Collection (of garbage)

As soon as the  
function gets  
executed, it  
gets wiped out of  
the Stack memory  
& console displays the output as required.

इसलिए बनता है  
+2 अब उसे  
कोई use नहीं  
करता।

Note:- Binary search गी यह करके भी  
Inverted Euler Tree हो जाता है  
+ यहाँ उसमें 2 फंक्शन हैं  
2 calls मिलती हैं।

# PRINT Zigzag <code>

```
→ import java.util.*;  
→ import java.io.*;  
→ public class Main {  
    public static void main(String args[]) {  
        Scanner scan = new Scanner(System.in);  
        int n = scan.nextInt();  
        pzz(n);  
    }  
}
```

//Expectation → pzz(3) → 3 2 ||| 2 ||| 2 3 2 ||| 2 ||| 2 3

//Faith → pzz(2) → 2

//Expectation meets Faith → 3 pzz(2) 3 pzz(2) 3

```
→ public static void pzz(int n) {
```

```
    if (n == 0)  
        return;  
    }
```

```
System.out.println(n + " pre");  
pzz(n - 1);
```

```
System.out.println(n + " in");  
pzz(n - 1);
```

```
System.out.println(n + " post");  
}
```

```
}
```

```
}
```

# Tower Of Hanoi

Tower Of Hanoi और Peint Zigzag ques के को सame हैं।  
Tower Of Hanoi की इस Peint Zigzag की application देखते ही  
इन दोनों के ques के Euler Tree का Use करते होते हैं।

1 → There are 3 towers.

Tower 1 has  $n$  disks, where  $n$  is a positive Integer

Tower 2 is empty, means 0 disks

Tower 3 is empty, means 0 disks.

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

3. → We are required to.

3.1. Peint the instructions to move the disks.

3.2. from tower 1 to tower 2 using tower 3.

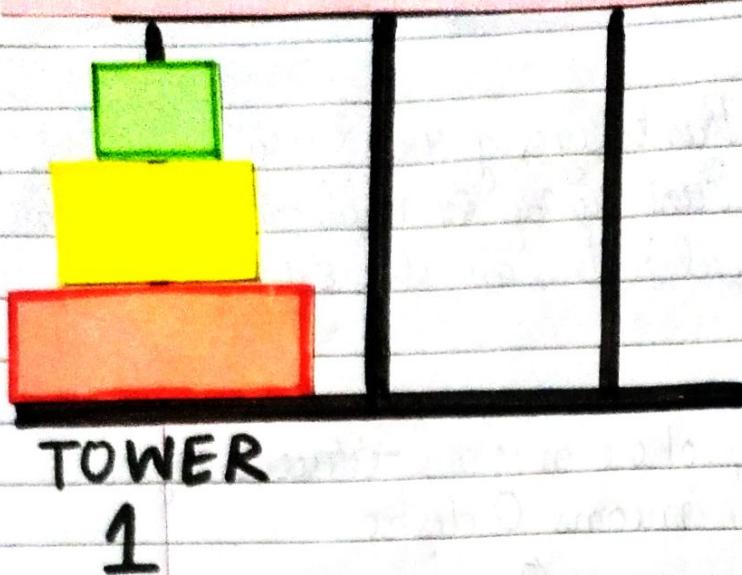
3.3. following the rules :-

3.3.1. move 1 disk at a time

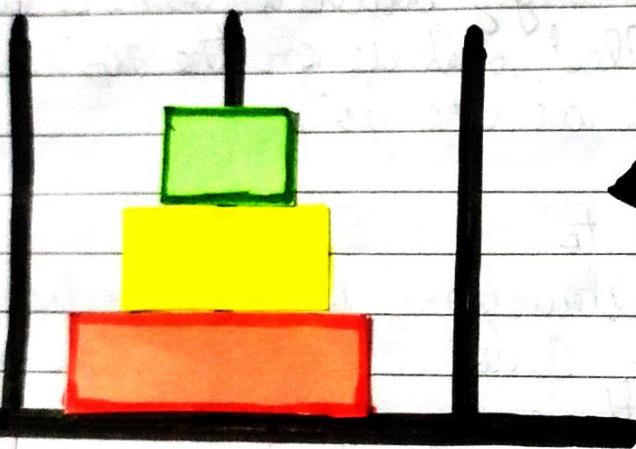
3.3.2. never place a smaller disk under a larger disk

3.3.3. You can only move a disk at the top

# Initial & Final Condition of Towers



← INITIAL  
CONDITION



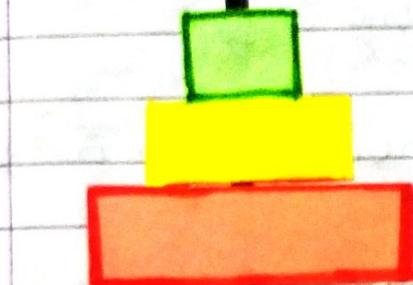
← FINAL  
CONDITION

1. Expectation  
HIGH LEVEL THINKING → 2. Faith

3. Expectation meets  
Faith

# High Level Thinking

Tower A  Tower B



Tower C

## 1. Expectation

$tah(3, A, B, C)$

↑  
all 3 disc

tower of hanoi

→ It is expected  
that the func  
 $tah(3, A, B, C)$   
needs to print  
the valid instruc  
tion to move  
3 disc from A to  
B using C

## 2. Faith

$tah(2, A, B)$

Print all the  
instructions following  
which we can  
move 2 discs from

A to C using B

→ दो मान

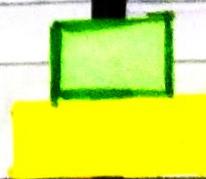
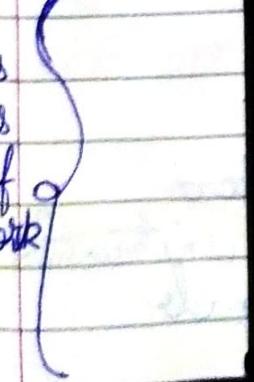
दो की 2

disc सही से

shift  
एक रुपा

Move Disc from A to B

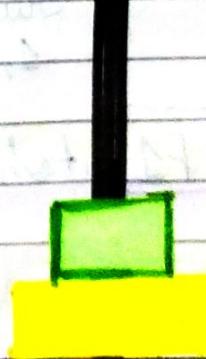
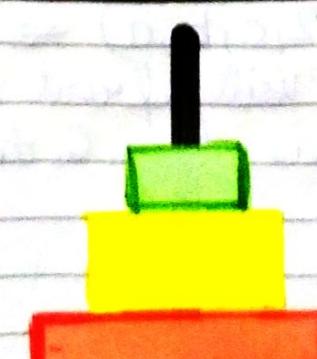
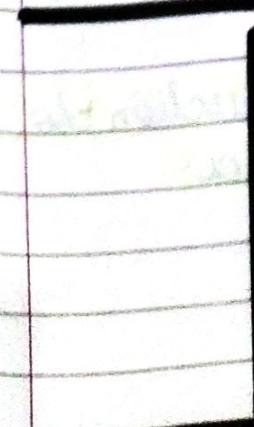
This  
is  
Self  
of  
work



## 3) Expectation meet Faith

$tah(2, C, B, A)$

Now, move  
2 disc from  
C to B using A



$tah(2, C, B, A)$

concepts understood

Page No.	
Date	

Now, using the High Level Thinking, we must stay to code

## Tower of Hanoi <code & dry run>

public static void main (String args)

{

```
Scanner scan = new Scanner (System.in);  
int n = scan.nextInt();  
int t1id = scan.nextInt();  
int t2id = scan.nextInt();  
int t3id = scan.nextInt();  
toh (n, t1id, t2id, t3id)  
n no. of disks  
↳ Name of first Tower  
↳ Name of second Tower  
↳ Name of 3rd Tower
```

Expectation  $\Rightarrow$  toh(3, a, b, c)  $\Rightarrow$  It is supposed to print valid (a/b/c) instructions to move 3 discs from a to b using c  
instruction following rule  
1, 2, 3

Faith = toh(2, a, cb)  $\Rightarrow$

It will print valid instruction to move 2 discs from a to c using b.

Faith 2,

= toh(2, c, b, a)  $\Rightarrow$

It will print valid instruction to move 2 discs from c to b using a.

t1id t2id t3id

public static void toh (int n, int t1id,  
int t2id, int t3id)

{

→ toh (n-1, t1id, t3id, t2id);

// now t1 will have Only 1 disc,  
t2 is now empty,  
t3 has 2 discs

t1id t2id t3id

System.out.println ( n + "[" + t1id +  
// t1 is empty, " → " +  
t2id +  
" ]");  
t3id  
t1id t2id t3id

→ toh (n-1, t3id, t2id, t1id);

// t1 is empty // t3 is empty // t2 has all the disk

t1id t2id t3id

public static void toh (int  $n$ , int  $t_1 id$ , int  $t_2 id$ , int  $t_3 id$ ) {

if ( $n == 0$ )  
{ return; }

toh ( $n - 1$ ,  $t_1 id$ ,  $t_3 id$ ,  $t_2 id$ ); → 1st time function calling

वह first call होगी तो function call करते हुए second और third towers की position exchange होगी। function के parameter में।

System.out.println ( $n + "[ " + t_1 id + " \rightarrow " + t_2 id + " ] "$ ),

↑  
In area.

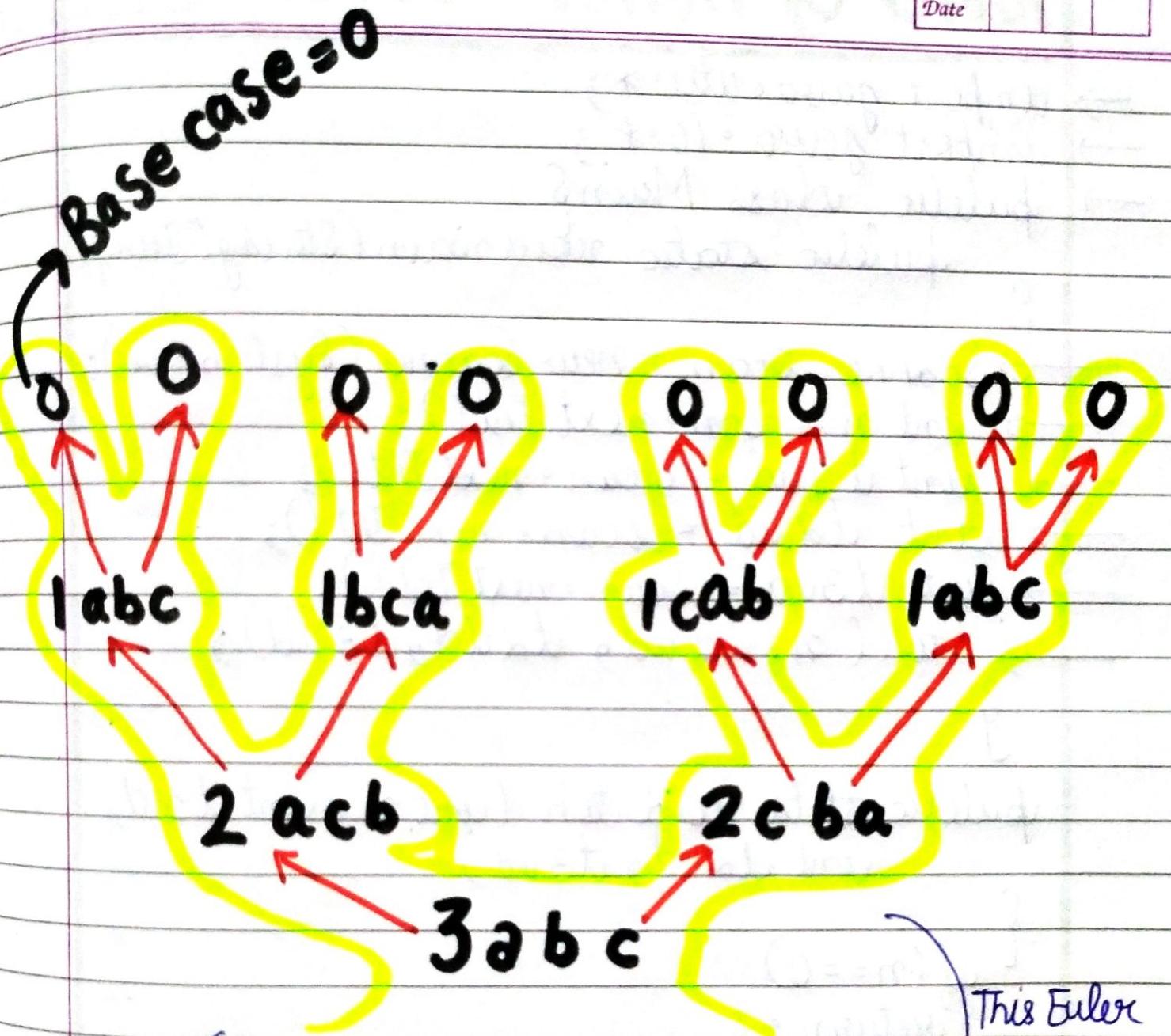
toh ( $n - 1$ ,  $t_3 id$ ,  $t_2 id$ ,  $t_1 id$ ); → 2nd time function calling.

वह second call होगी तो function call करते हुए first और third towers की position exchange होगी। function के parameter में।

↑  
No Post Area.

}

अब ये follow करते हुए Euler tree बनाने का आया जाएगा।



(Output)

$$\begin{aligned}
 1 &\rightarrow [a \rightarrow b] \\
 2 &\rightarrow [a \rightarrow c] \\
 1 &\rightarrow [b \rightarrow c] \\
 3 &\rightarrow [a \rightarrow b] \\
 1 &\rightarrow [c \rightarrow a] \\
 2 &\rightarrow [c \rightarrow b] \\
 1 &\rightarrow [a \rightarrow b]
 \end{aligned}$$

This Euler Tree traversal generates the following output

# Tower Of Hanoi <code>

Page No.

Date

```
→ import java.util.*;
→ import java.io.*;
→ public class Main {
    public static void main (String [] args)
    {
        Scanner scan = new Scanner (System.in);
        int n = scan.nextInt();
        int t1id = scan.nextInt();
        int t2id = scan.nextInt();
        int t3id = scan.nextInt();
        toh (n, t1id, t2id, t3id);
    }
}
```

```
public static void toh (int n, int t1id,
                      int t2id, int t3id)
```

```
{
    if (n == 0)
        { return; }
}
```

```
→ toh (n-1, t1id, t3id, t2id);
→ System.out.println (n + "[" + 
                     t1id + " -> " + t2id
                     + "]");
→ toh (n-1, t3id, t2id, t1id);
```

```
}
```

```
3.
```

Introduction  
To

Recursion

DONE

— Samanna Verma