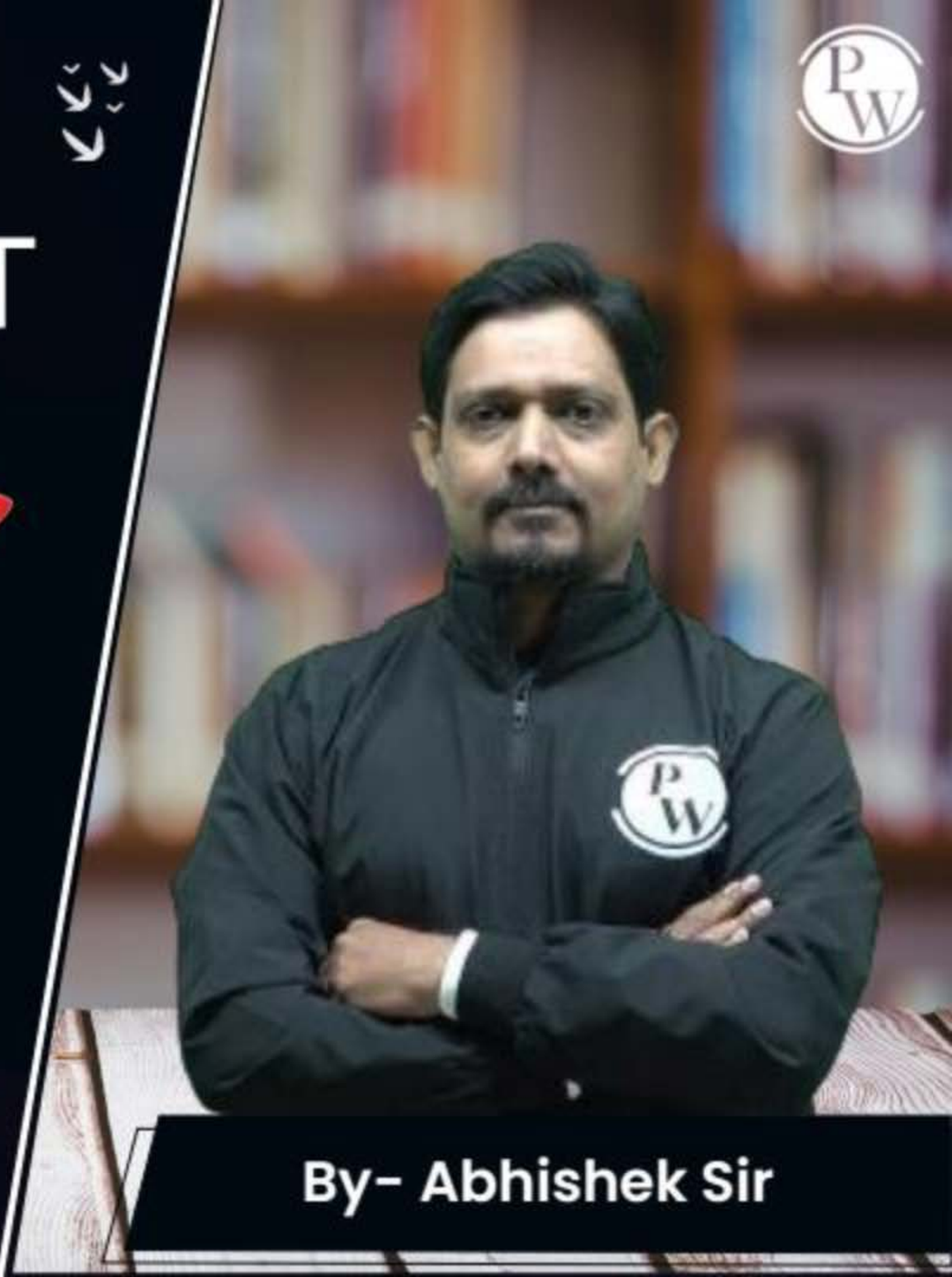
Computer Science & IT

C Programming

Function & Storage Class

Lecture No. 04



By- Abhishek Sir

Recap of Previous Lecture

- Topic
- Topic — Recursion
- Topic — Tail / Non tail
- Topic
- Topic
- Topic

Slide

Topics to be Covered

- Topic — TOH
- Topic — Recursion Nested
- Topic — Indirect Recursion
- Topic
- Topic

Slide

Tower of Hanoi

1 There are 3 Towers / pegs are given L, M, R (Left, Middle, Right)

2 There are n disc of different size

3 All n disc are placed in tower L such that Smallest disc will be on top. (Large disc will not be top of smaller disc.)

Slide

Tower of Hanoi

4 We need to move all n disc from tower L to tower R using tower M.

5 Only one disc can be moved at a time.

Slide

Tower of Hanoi

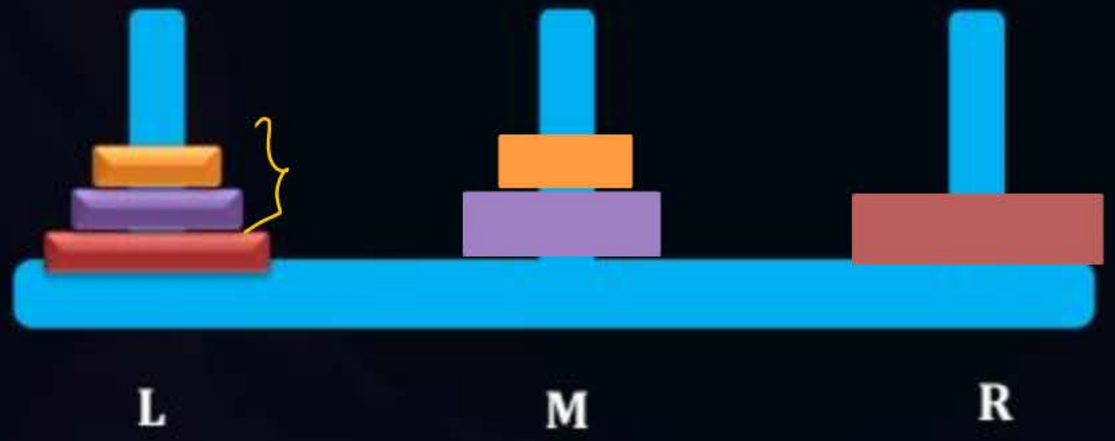
TOH (n, L, M, R)

L — Aux — D

we required to move n disc from tower L to tower R using tower M

n-1 disc Subproblem

TOH (n-1, L , R , M)



Tower of Hanoi

Tower of Hanoi

TOH(n, L, M, R)

TOH(n-1, L, R, M),

Move the disc from L → R

Tower of Hanoi

TOH (n, L, M, R)

TOH(n-1, L, R, M);

Move the disc from L → R ,

TOH (n-1, M, L , R);

```
Algorithm TOH(n, L, M, R) {
                1 2 3
    if(n>=1) {
                    1 3 2
        TOH(n-1, L, R, M),
        Move the disc from L→R,
                    2 1 3
        TOH(n-1, M, L, R),
    }
}
```

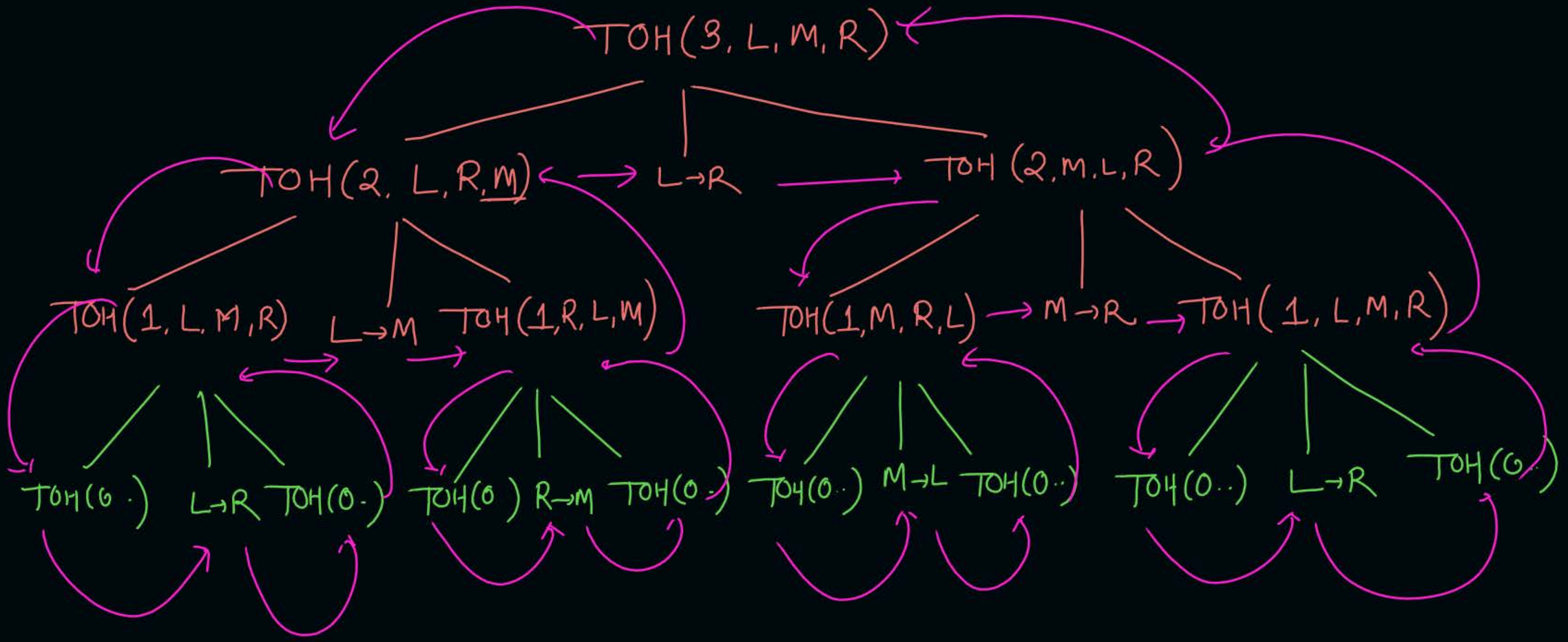
disc Movement

Establish Recurrence Relation

T(n) represent No. of disc movement

$$T(n) = 2T(n-1) + 1 \quad n \geq 1$$

$$T(n) = 0 \quad \text{if } n = 0$$


TOH(3, L, M, R)
TOH(2, L, R, M)
L→R
TOH(2, M, L, R)
TOH(1, L, M, R)
L→M
TOH(1, R, L, M)
TOH(1, M, R, L)
M→R
TOH(1, L, M, R)
TOH(0..)
L→R
TOH(0..)
TOH(0..)
R→M
TOH(0..)
TOH(0..)
M→L
TOH(0..)
TOH(0..)
L→R
TOH(0..)

$$T(n) = 2T(n-1) + 1$$

Sowe

$$2^n - 1$$

$$T(0) = 0$$

$$T(1) = 2T(0) + 1 = 1$$

$$T(2) = 2T(1) + 1 = 2 \times 1 + 1 = 3$$

$$T(3) = 2T(2) + 1 = 2 \times 3 + 1 = 7$$

Tower of Hanoi

Tower of Hanoi

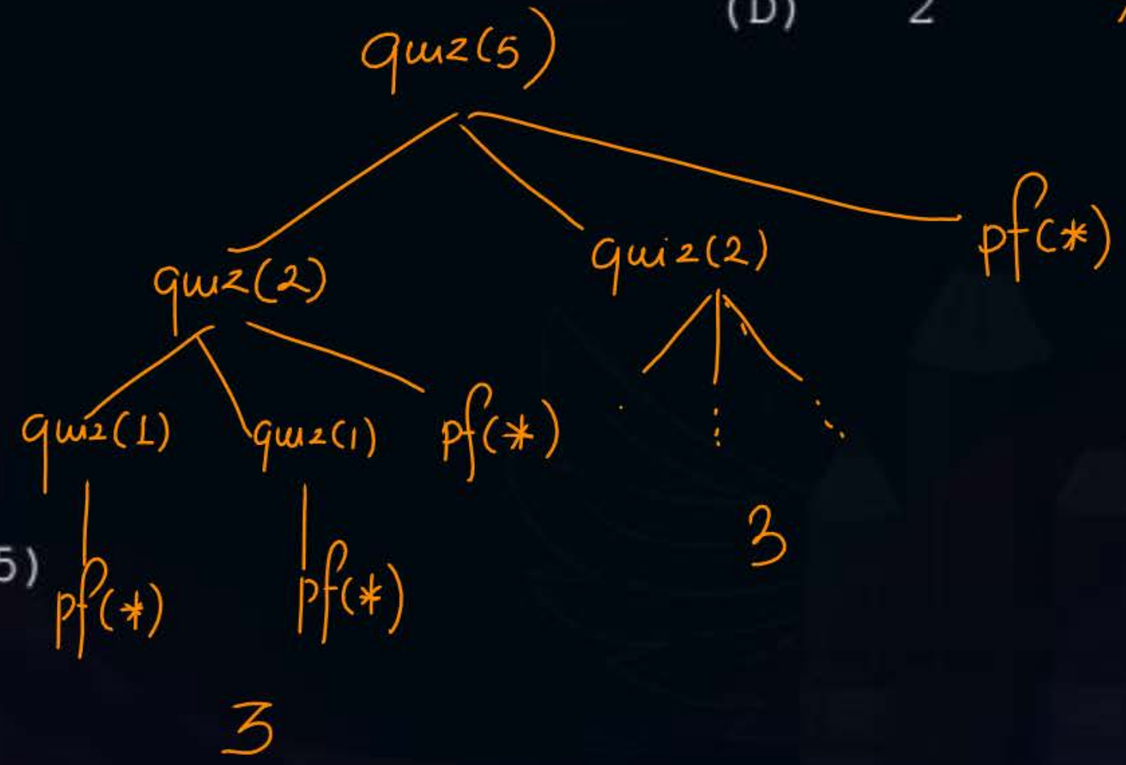
Recursion

#Q. Consider the following program

```
void quiz(int i)
    {
        if (i > 1)
        {
            quiz(i / 2);
            quiz(i / 2);
        }
        printf("*");
    }
Number of star printed quiz(5)
```

(A) 8
(B) 7
(C) 3
(D) 2

7



Slide

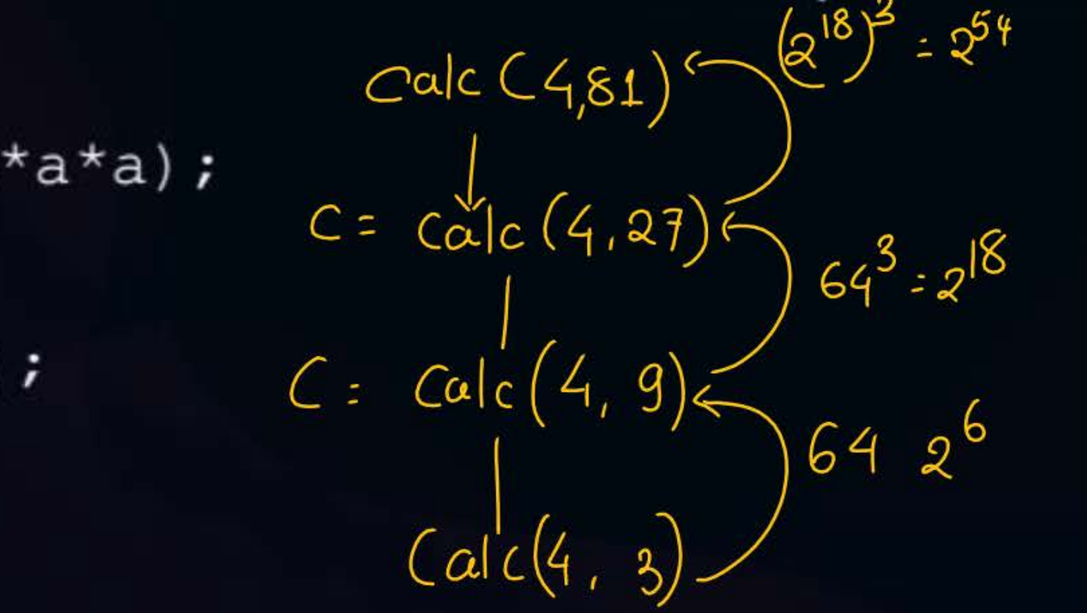
Question

Consider the following C program:

PYQ

```
#include <stdio.h>
int counter = 0;
int calc (int a, int b) {
int c;
Counter ++;
if (b==3) return (a*a*a);
Else {
   c = calc (a, b/3);
   return (c*c*c);
   }
}
int main () {
   calc (4,81);
   print f ("%d", counter);
}
```

The output of this program is 4



Consider the following program written in pseudo-code. Assume that x and y are integers.

```
Count (x,y){
If (y != 1) {
    if (x! = 1) {
      print f("x");
      Count (x/2, y);
    }
    else {
          y = y - 1
          count (1024, y);
        }
     }
}
```

Count (1024, 1024)

The number of times that the print statement is executed by the call count is ___

Question

10 (1024 — 2) 1024 - 2 + 1 = 1023

Consider the following program written in pseudo-code. Assume that x and y are integers.

```
Count (x,y){
If (y != 1) {
    if (x! = 1) {
      print f("x");
      Count (x/2, y);
    }
    else {
          y = y - 1
          count (1024, y);
       }
    }
}
```

count(1024,1024)
x y
1 count(512,1024)
2 count(256,1024)
3 count(128,1024)
4 count(64,1024)
5 count(32,1024)
6 count(16,1024)

count(16,1024)
7 count(8,1024)
8 count(4,1024)
9 count(2,1024)
10 count(1,1024)
Count(1024,1023)
Count(1024,2)
Count(1,2)

10230

The number of times that the print statement is executed by the call count is

Question

Consider the following recursive C function.

```
#include <stdio.h>
int fun(int x){
    if (x == 0)
        return 0;
    else
    if (x >4)
        return x;
    else
    return fun(10+fun(2*x));
}

int main(){
printf("%d", fun(fun(2)));
return 0;
}
```

(A) 15 (B) 28 (C)35 (D) 45

if Recursive call is parameter to function then we call that Recursion as Nested Recursion

fun(fun(2))

fun(28)

fun(2)

fun(10+fun(4)) (18)

fun(10+fun(8))

8

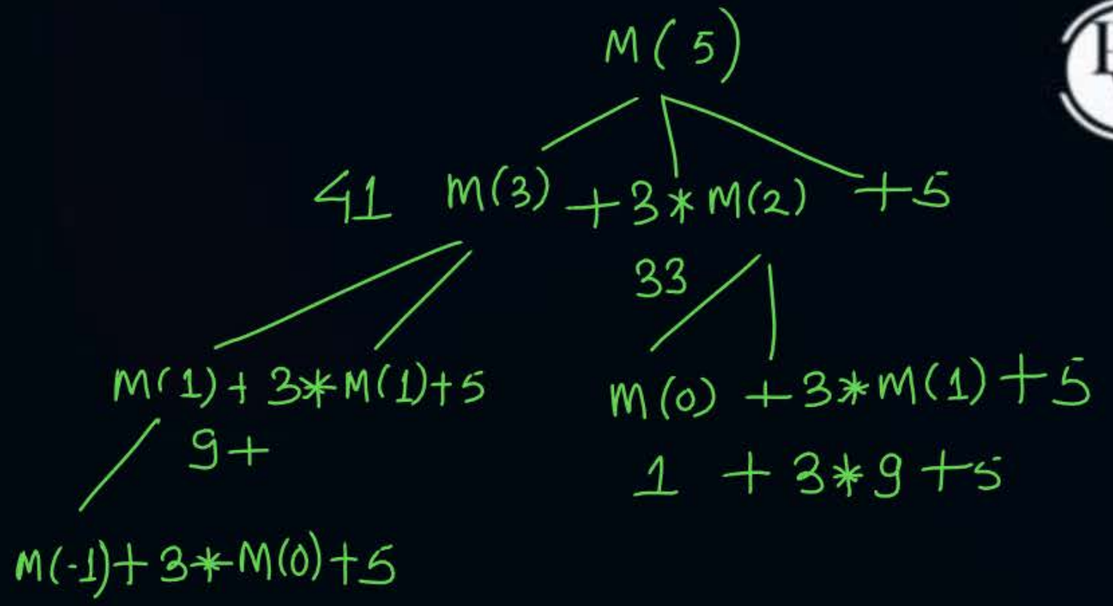
Slide

Question

Consider the following recursive C function.

```
int mystery(int n) {
        int answer;
        if (n > 0) {
        answer =(mystery(n-2)+3*mystery(n/2) + 5);
                return answer;
        }
        else
        return 1;
}
```

The return value of `mystery(5)` is _____.



Slide

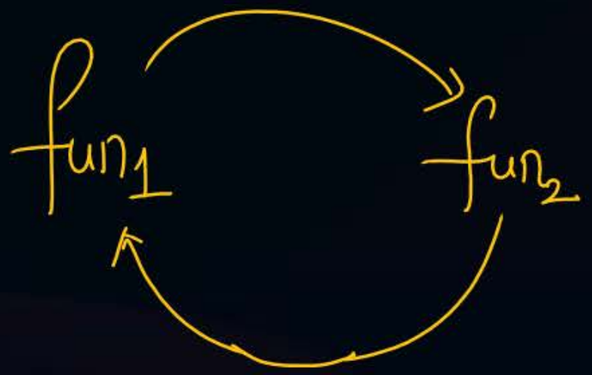
Indirect Recursion

```
void fun1 (int n) {
if (n = =0 ) return;
printf ("%d" , n);
fun2 (n - 2);
}
int main(){
Fun1(3);
}
```

```
void fun2 (int n) {
if (n = = 0) return ;
printf ("%d" , n);
fun1(++n) ;
}
```

Counter 4

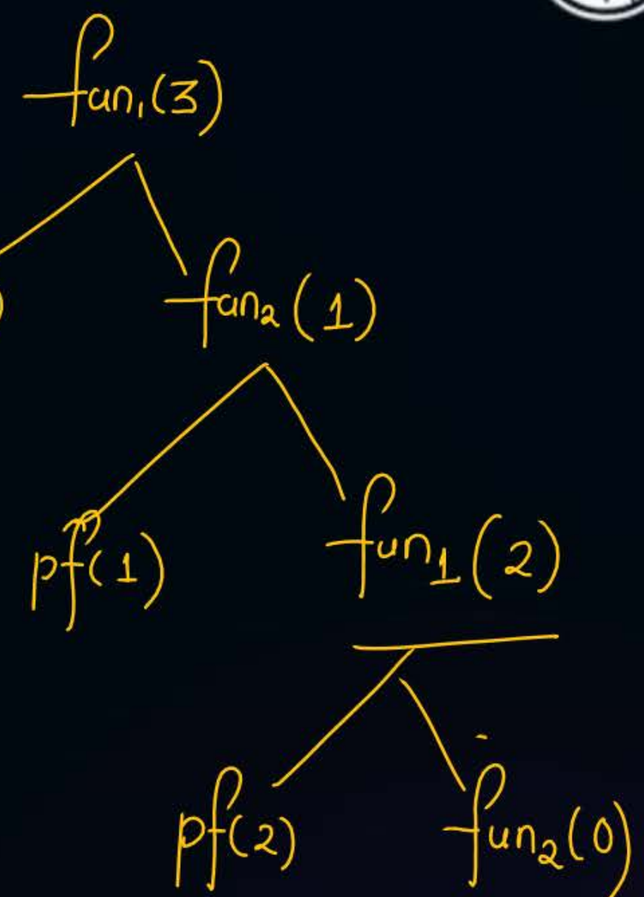
In Indirect Recursion more than 1 function is present and they call each other in circular manner



Indirect Recursion

```
void fun1 (int n) {
if (n = =0 ) return;
printf ("%d" , n);
fun2 (n - 2);
}
int main(){
Fun1(3);
}
```

```
void fun2 (int n) {
if (n = = 0) return ;
printf ("%d" , n);
fun1(++n) ;
}
```



If No. of values printed by $fun_1(4)$ is X and No of values printed by $fun_2(3)$ is Y then $X + 10Y$ is ____

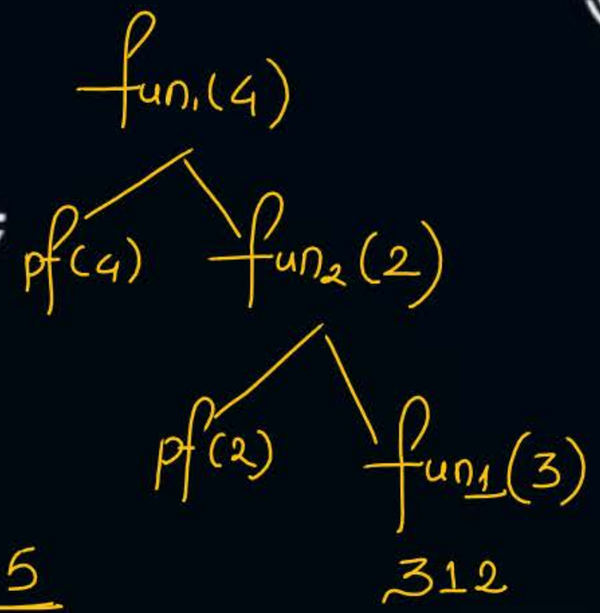
X = 5
Y = 6

5 + 10 × 6
= 65

Indirect Recursion

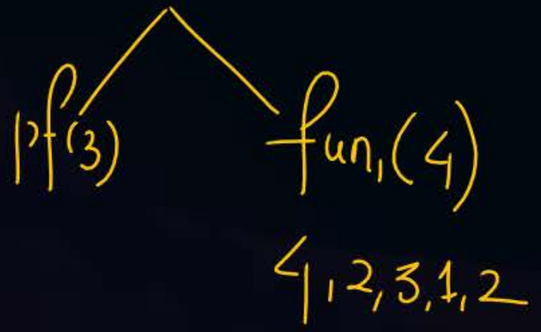
```
void fun1 (int n) {
if (n = =0 ) return;
printf ("%d" , n);
fun2 (n - 2);
}
int main(){
Fun1(3);
}
```

```
void fun2 (int n) {
if (n = = 0) return ;
printf ("%d" , n);
fun1(++n) ;
}
```



$fun_1(4) = 4,2,3,1,2 = 5$

$fun_2(3) = 3,4,2,3,1,2$



```
while (v) {
    count++ = v & 1,
    v >> 1,
}
```

count = 1

count = 2

v = 0101 ← LSB

& 0001 ←

0001

v 0010

& 0001

0000

Bitwise AND

v >> 1 0001 ← LSB

0001

0001 ←

v >> 1 0000

while loop end

2 mins Summary

Topic — TOH

Topic — Nested Recursion

Topic — Indirect Recursion

Topic

Topic

Slide

THANK - YOU