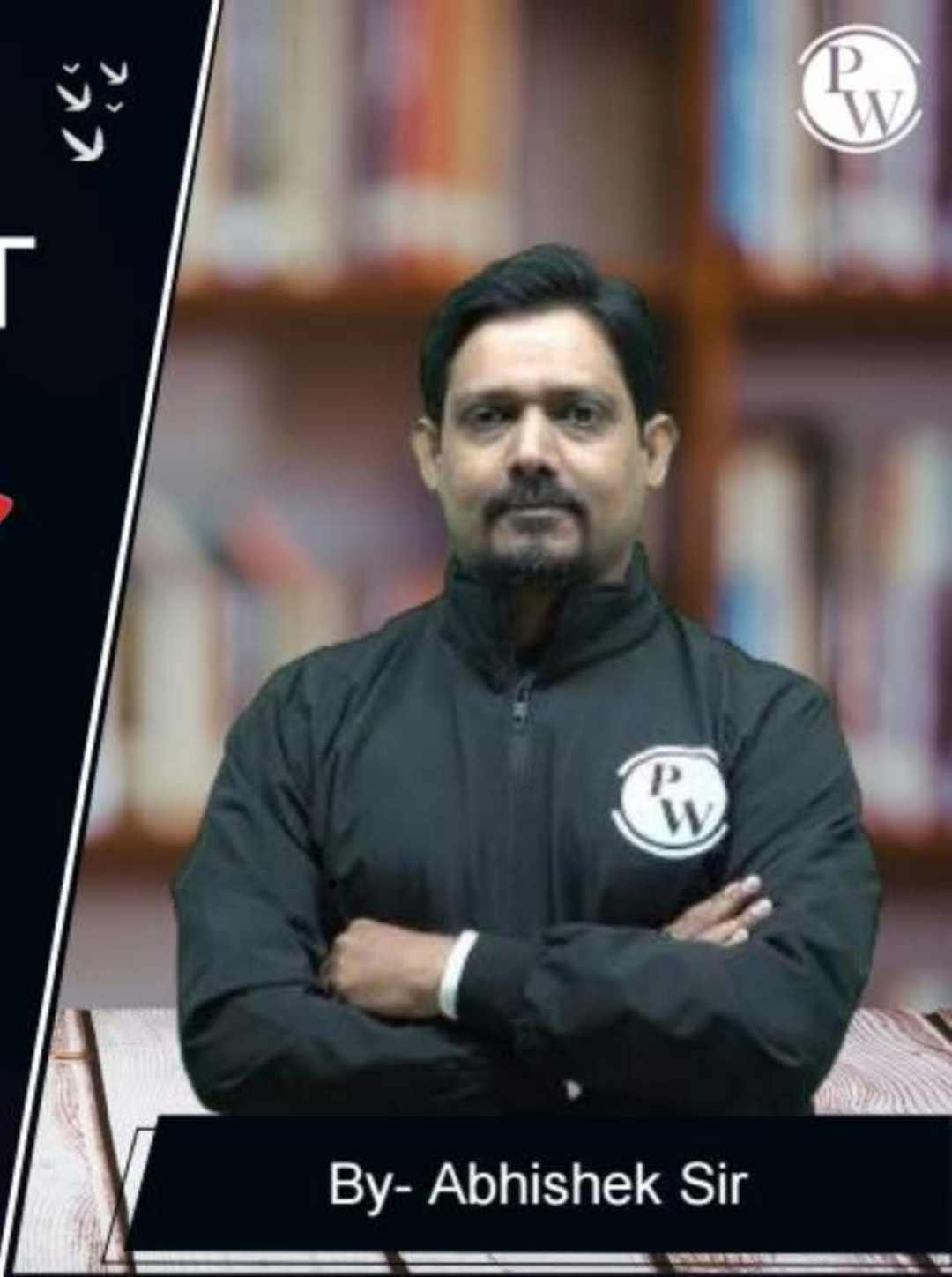
Computer Science & IT

C programming

Function & Storage Class

Lecture No. 05



By- Abhishek Sir

Recap of Previous Lecture

Topic — Recursion practice

Topic — Nested Recursion

Topic — Indirect Recursion

Topic

Topic

Slide

Topics to be Covered

- Topic
- Topic
- Topic
- Topic
- Topic

Towers of Hanoi ✓

Extern & Register ✓

pointers

Slide

Homework

Consider the following recursive C function.

```
void get(int n){
     if (n<1) return;
     get (n-1);
     get (n-3);
     printf("%d", n);
}
```

counting

2015

Invoked ≡ function Call

If function get(6) is being called in main() then how many times will the get function be invoked before returning to the ?

(A) 15 (B) 25 (C) 35 (D) 45

Slide

Homework

Consider the following recursive C function.

```
void get(int n){
     if (n<1) return;
     get (n-1);
     get (n-3);
     printf("%d", n);
}
```

counting

2015

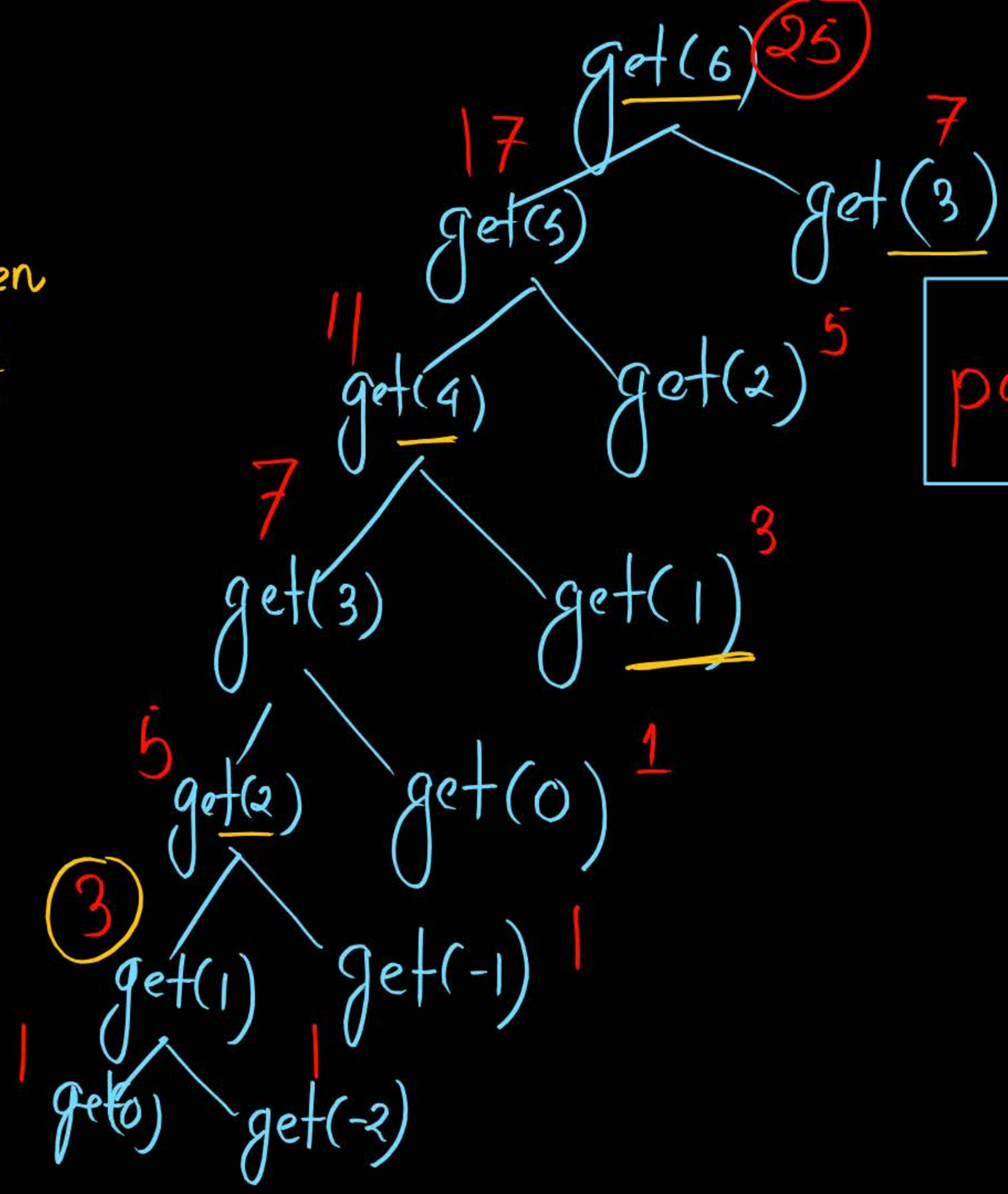
Invoked ≡ function call

If function get(6) is being called in main() then how many times will the get function be invoked before returning to the ?

(A) 15 (B) 25 (C) 35 (D) 45

Slide

Two children
one parent



partial Recursion Tree

Counting & print
partial Recursion

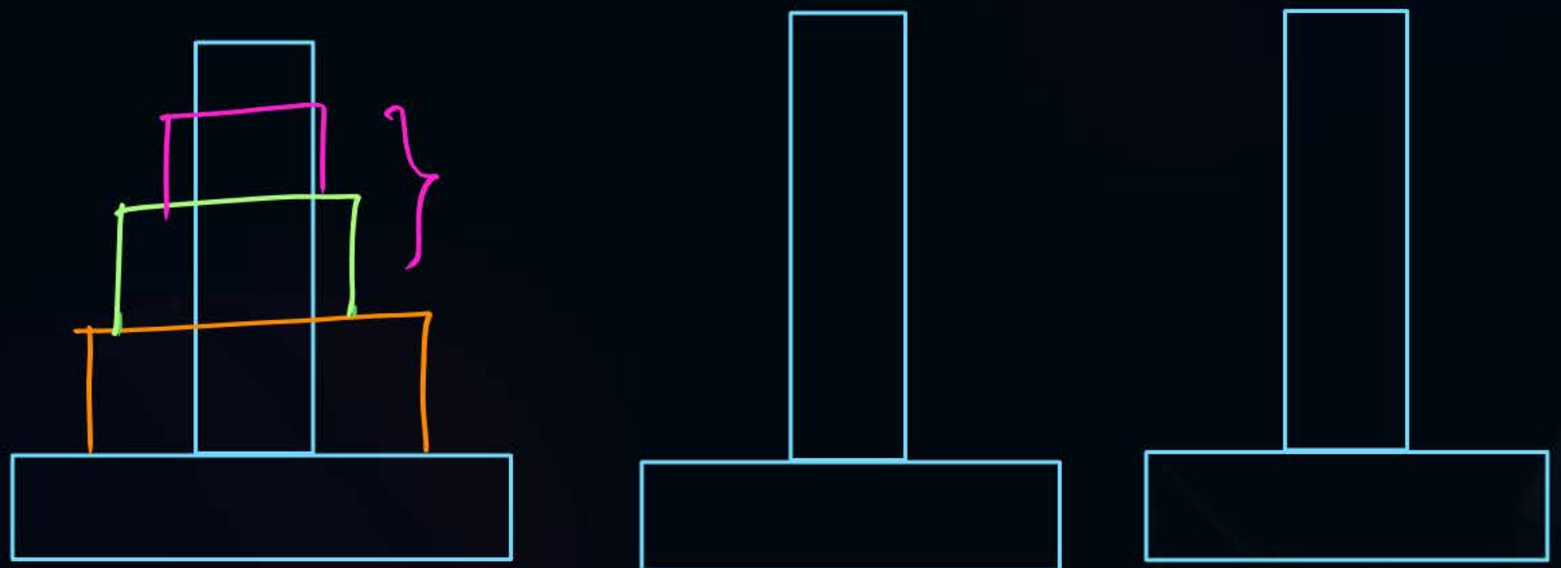
Tower of Hanoi

No Large disk will be on top of Smaller disk

* There 3 towers Left(L), middle(M), Right(R)
* n disk of different size is given
* All n disk are placed in tower(L) in Decreasing order of their size
* we want to move all n disk from tower L to tower R using tower M. Such that No large disk will be on top of Smaller disk

Slide

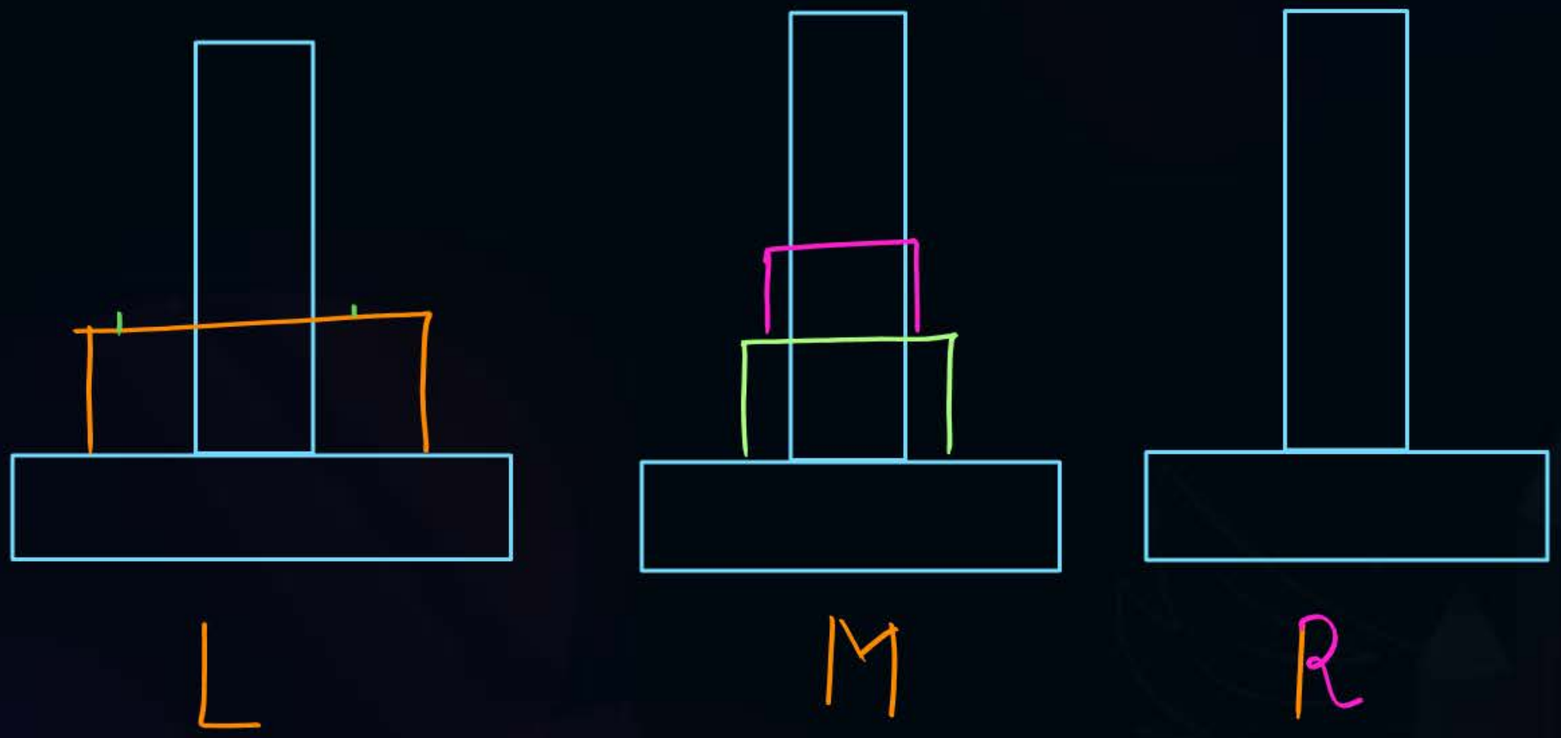
Tower of Hanoi



The bottom disk in R will be

Slide

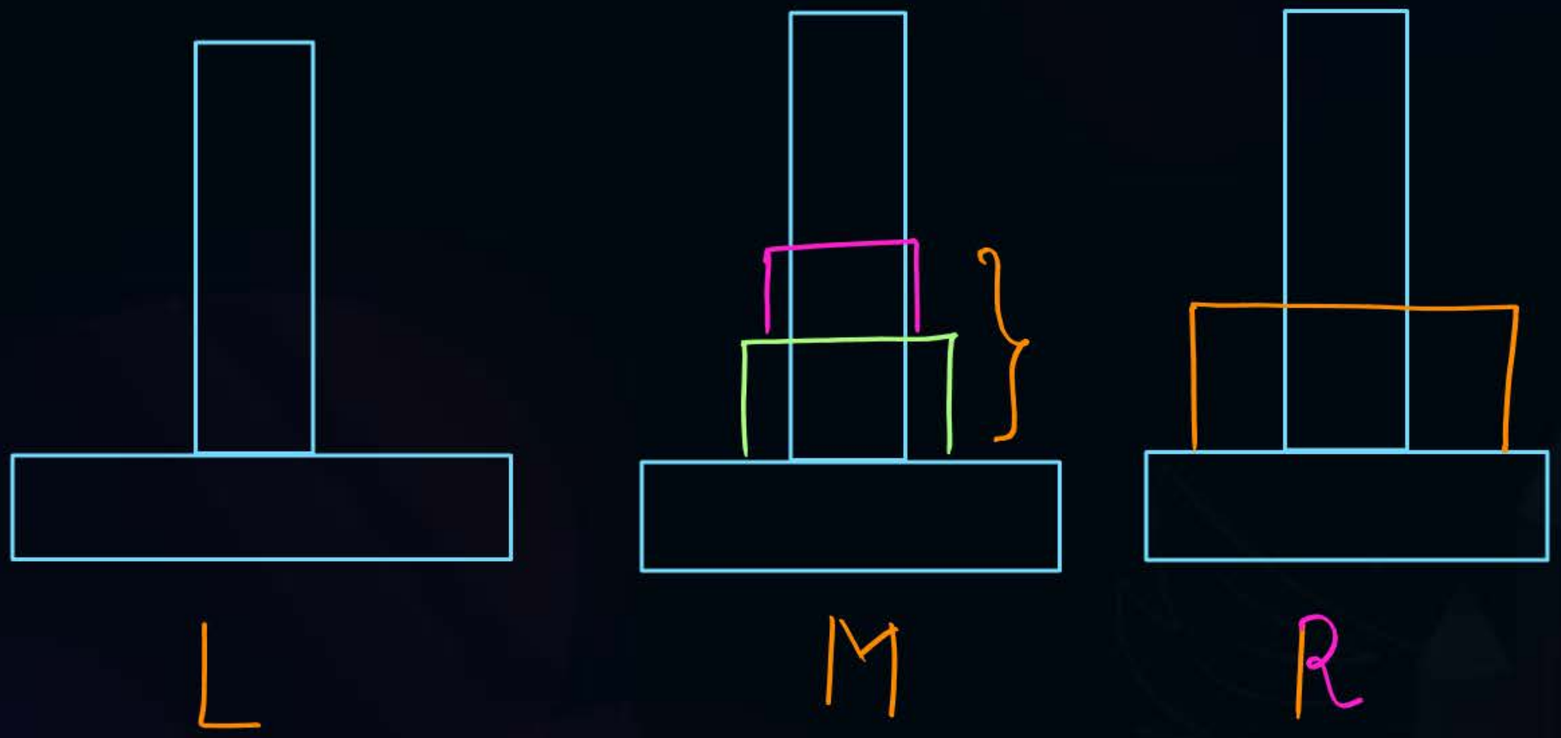
Tower of Hanoi



The bottom disk in R will be

Slide

Tower of Hanoi



The bottom disk in R will be

Slide

Tower of Hanoi

Source Destinal

Algorithm TOH(3, L, M, R) { ← Move n disk from L → R using M

TOH(2, L, R, M);

Move the bottom disk from L → R ✓

TOH(2, M, L, R);

}

Slide

Tower of Hanoi

1 2 3

Algorithm TOH(n, L, M, R) { ← Move n disk from L → R using M

if(n >= 1) { 1 3 2

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

1 → 3

Move the bottom disk from L → R ← disk movement

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

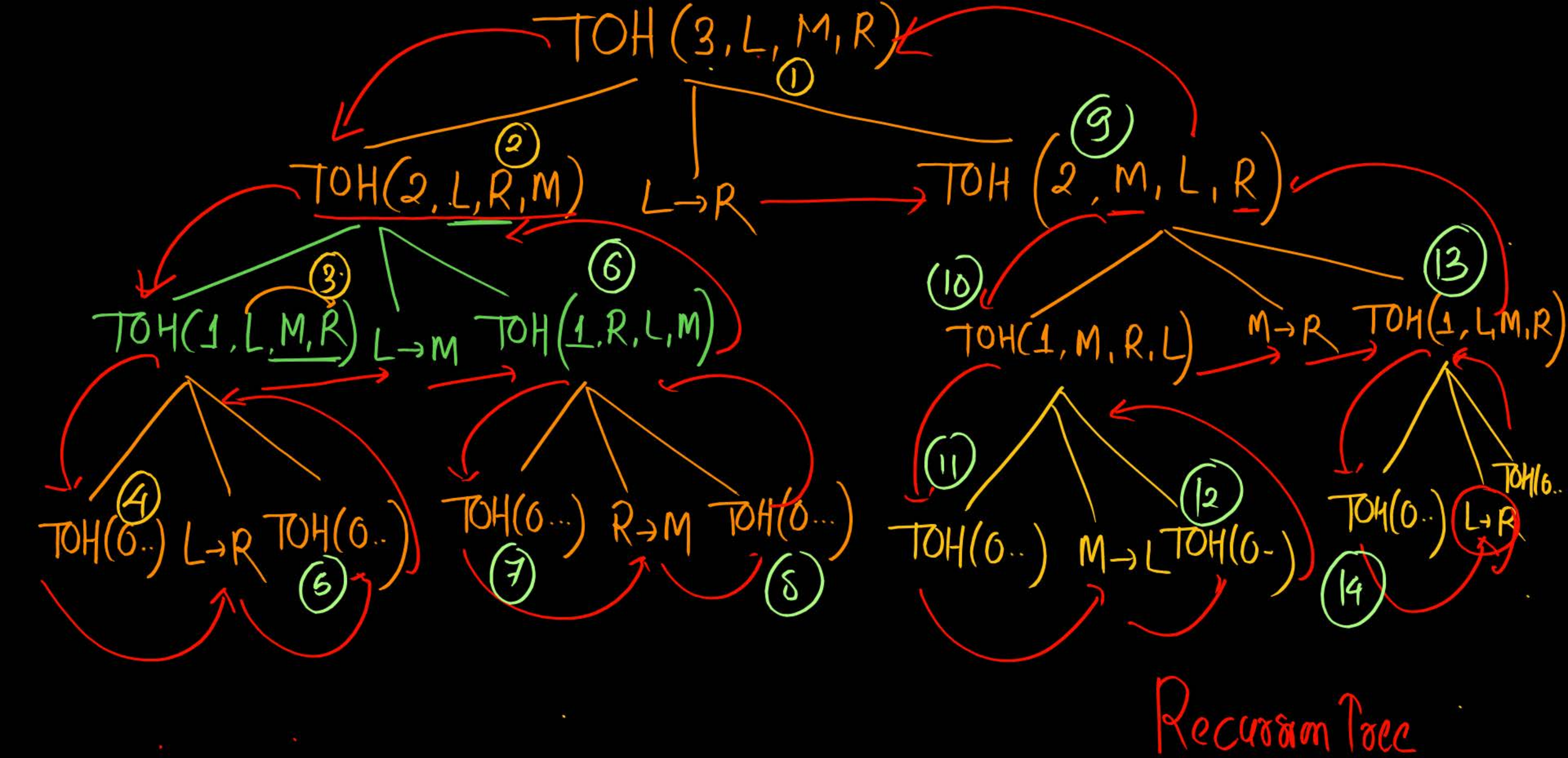
2 1 3

}

}

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

Slide


TOH(3, L, M, R)
1
TOH(2, L, R, M)
2
L→R
TOH(2, M, L, R)
9
TOH(1, L, M, R)
3
L→M
TOH(1, R, L, M)
6
TOH(1, M, R, L)
10
M→R
TOH(1, L, M, R)
13
TOH(0...)
4
L→R
TOH(0...)
5
TOH(0...)
7
R→M
TOH(0...)
8
TOH(0...)
11
M→L
TOH(0...)
12
TOH(0...)
14
L→R
TOH(0...)
Recursion Tree

Tower of Hanoi

10. If the number of disks are 3 in **Towers of Hanoi**
    (a) After how many Invocations, there is a First move 4 — function call
    (b) After how many Invocations, there is a Last move _____
    (c) Total moves _____ 7
    (d) Total Invocations 15

Substitution Method

function

if T(n) represent No. of disk movement

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

$2^n - 1$

No. of disk movement with n disk $= 2^n - 1$

$2^3 - 1 = 7$

n = 4 disk.

$2^4 - 1 = 15$

Tower of Hanoi

Tower of Hanoi

Tower of Hanoi

Tower of Hanoi

Tower of Hanoi

Tower of Hanoi

Register Storage class

1 Auto: defaut storage class
- for Local (allocate, deallocate)

Storage: Stack

Life : until-funchon is achve.

2. Static (Local 2 global)

Storage : Staticdata

Liftime
until program
is running.

Visibiley
within funchon

visibiley
evey funchon

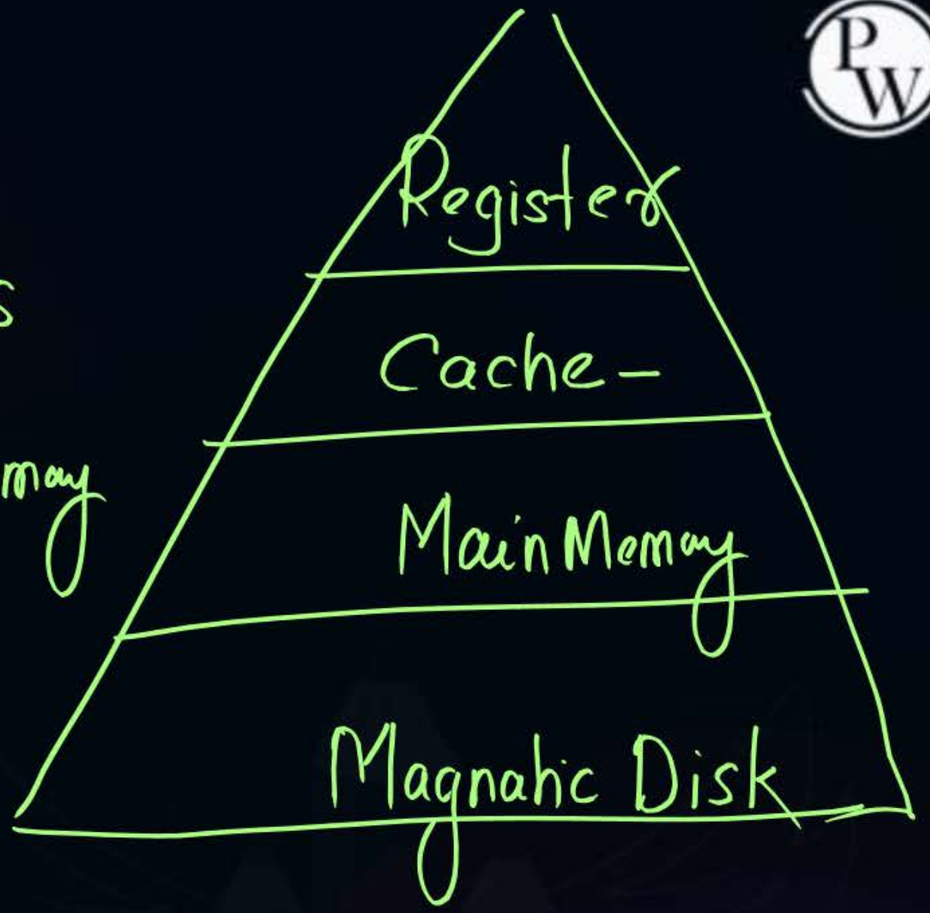
Initialized data segment
↑ New Name

Slide

Register Storage class

Register storage class : processor's Nearest memory

Some important variable for faster access register storage class is used



Slide

Register Storage class

* Size of data type restricted by Register Size.

* & Address of operator Cant be applied.

register int a;

Slide

Extern Storage Class

1. Extern storage extends visibility of variable
2. Difference between declaration & Definition

When a variable is declared the we announce property of variable. No memory allocated.

```
int a;
```

Memory is allocated & if assigned value is stored.

Slide

Extern Storage Class

3. extern int a; ← Declaration Not definition

No memory allocation.

It simply ask compiler to extend the visibility may be outside of brack ({ }) or outside of program as well.

4. Big project uses lot of variable hence a seperate file created for variable only and extern keyword is used to access them.

Slide

Extern Storage Class

5. If variable is No-found then 'Linker error' is generated

Pointer

pointer + Array + Linked List

int, char, float Data type

pointer is also a Data type

pointer is a variable that can hold address of similar Data type

pointer holds address

pointer r-value is an Address.

Slide

Pointer

Declaration of pointer variable

int *ptr; ptr is pointer to an integer

Initialization of pointer variable

int a=10;

int *ptr=&a;

* dereference operator

& address of operator.

Slide

Pointer

Assignment of pointer variable

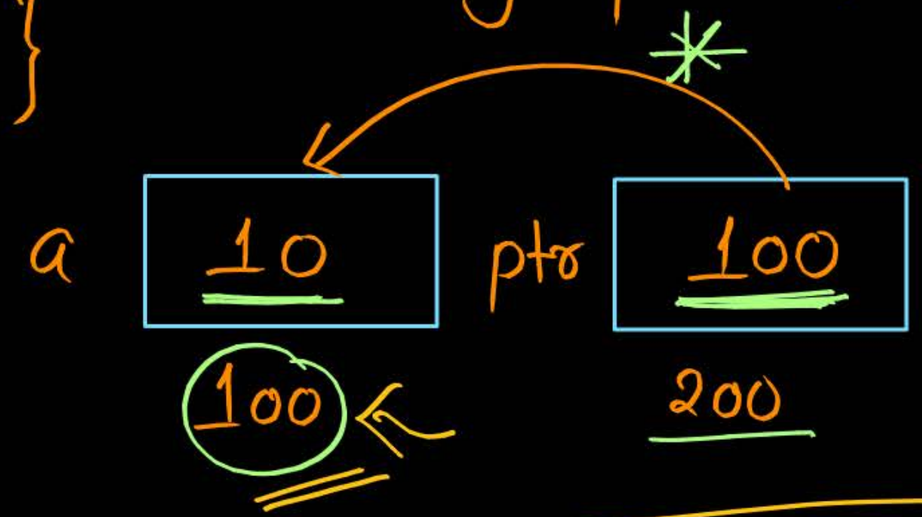
```
int a = 10;
int *ptr;
ptr = &a;
```

Slide

```c
#include<stdio.h>
int main(){
    int a=10;
    int *ptr=&a;
    printf("%d", a);      10
    printf("%u", ptr);    100
    printf("%d", *ptr);   10
    printf("%u", &a);     100
}
```

```c
printf("%u", &ptr);  200
```

jump u.

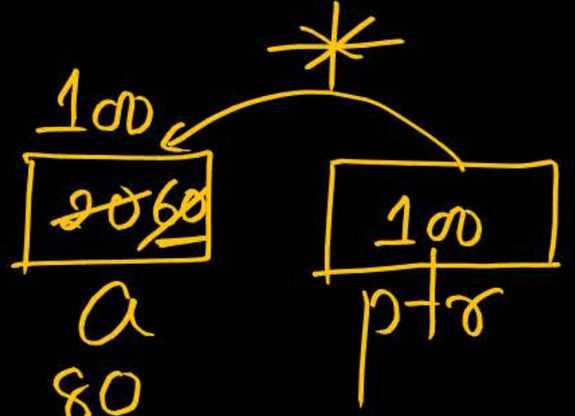


%p : Hexdacimal

%u

(A) 60
(B) 80
(C) 20
(D) 40

```
#include<stdio.h>
int main(){
    int a=20;
    int *ptr;
    ptr=&a;
    a=a+40;
    *ptr=*ptr+20;
    printf("%d", a);
}return 0;
```



*ptr+20

60+20

80

H.W

Gate 2015

for loop value calculate

Recursion

1. u ← Address

No. of disk Movement

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

$T(6)$

$= T(1) = 1$

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

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

$T(4) = 2 \times 7 + 1 = 15$

$T(5) = 2 \times 15 + 1 = 31$

$T(6) = 2 \times 31 + 1$

$= 63$

2 mins Summary

- Topic
- Topic
- Topic
- Topic
- Topic

Tower of Hanoi

Extern Register overview

pointer

preprocessor

Replace Sum by 100

Slide

THANK - YOU