unacademy

Trees - Part III

Course on Data Structure

Pankaj Sharma • Lesson 23 • Aug 2, 2024

unacademy

CS & IT Engineering

Data Structure

Tree

Lecture Number- 23

By- Pankaj Sir

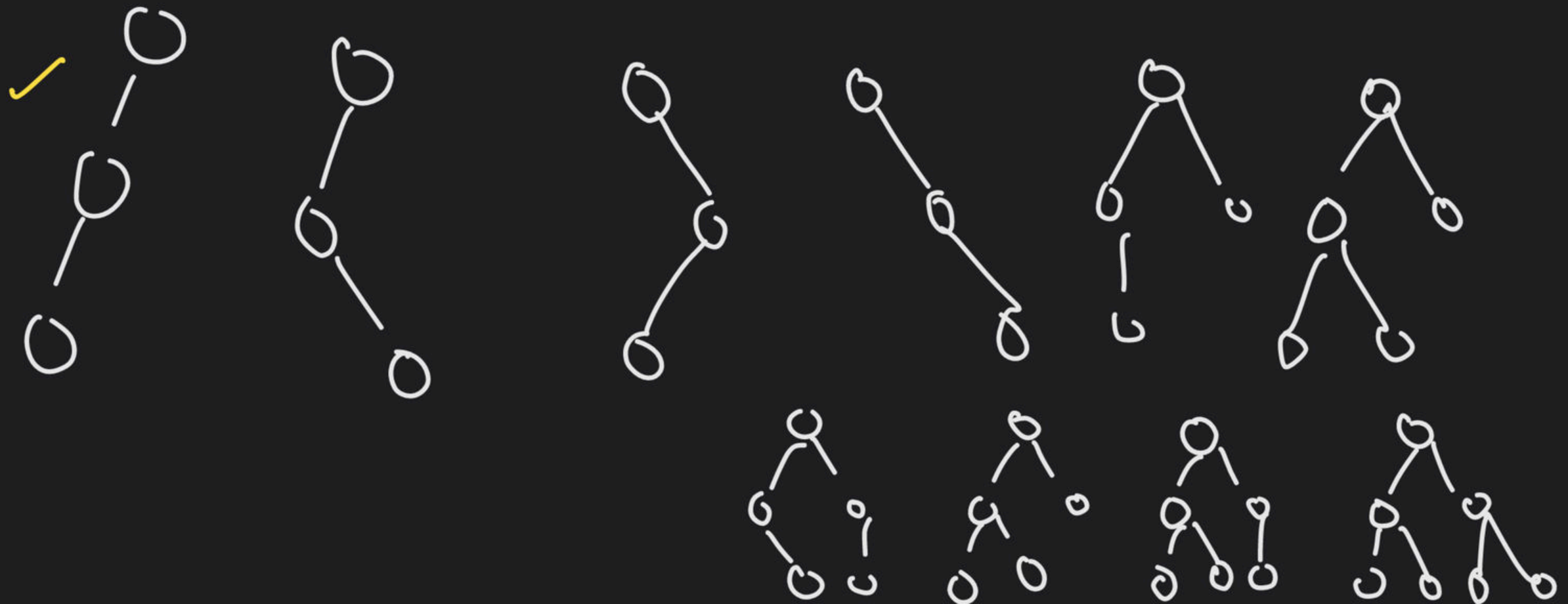
unacademy

Topics

to be covered

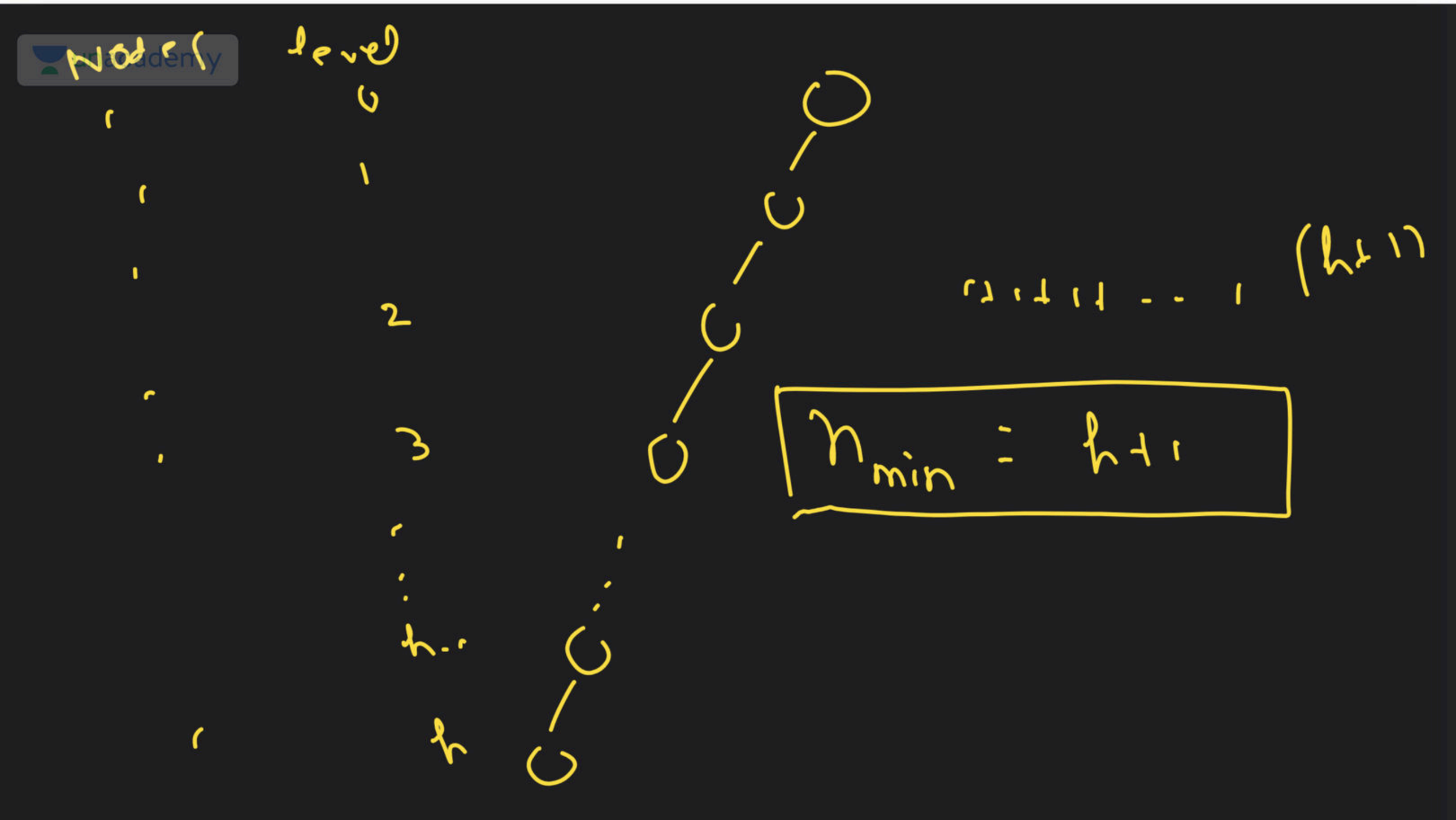
1. Tree-II

Min no. of nodes in a binary tree of height h.

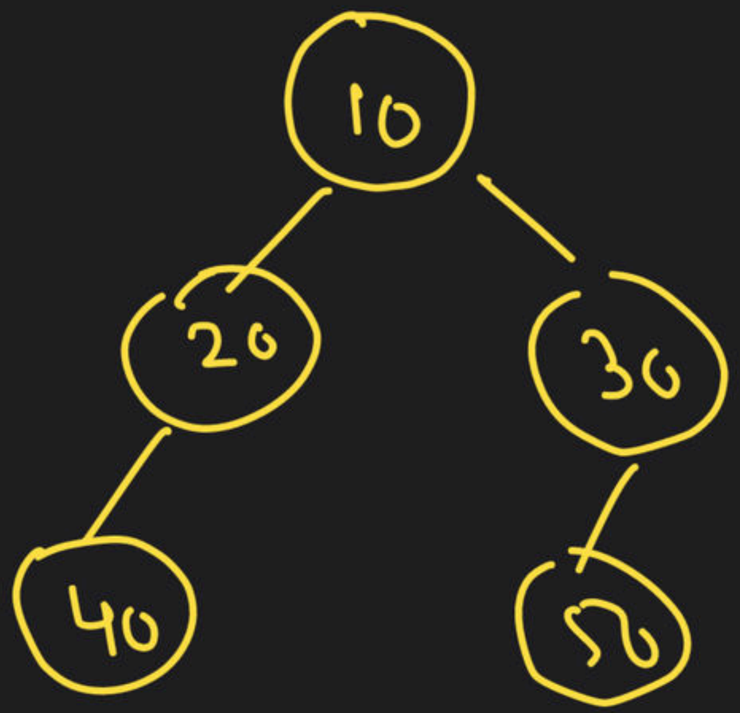


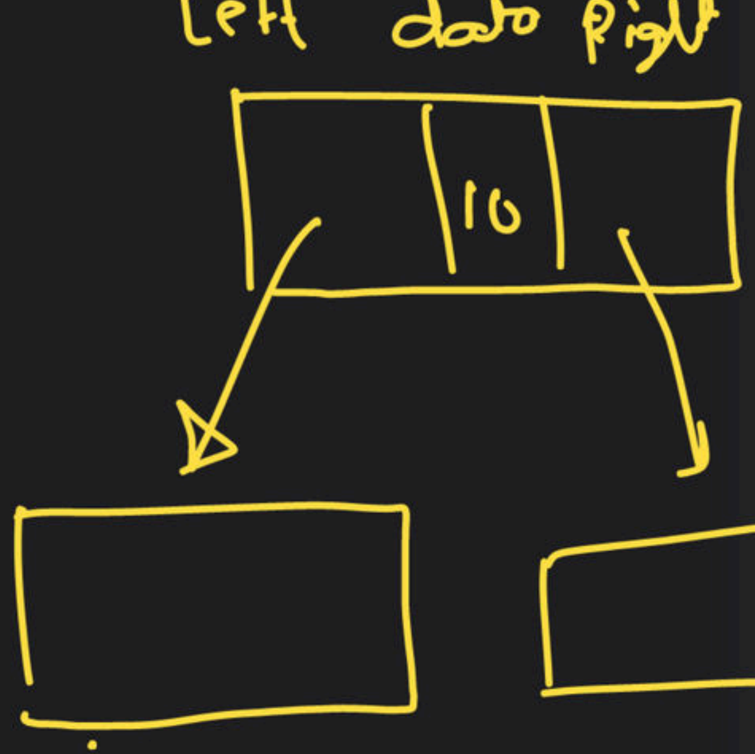
h : 2

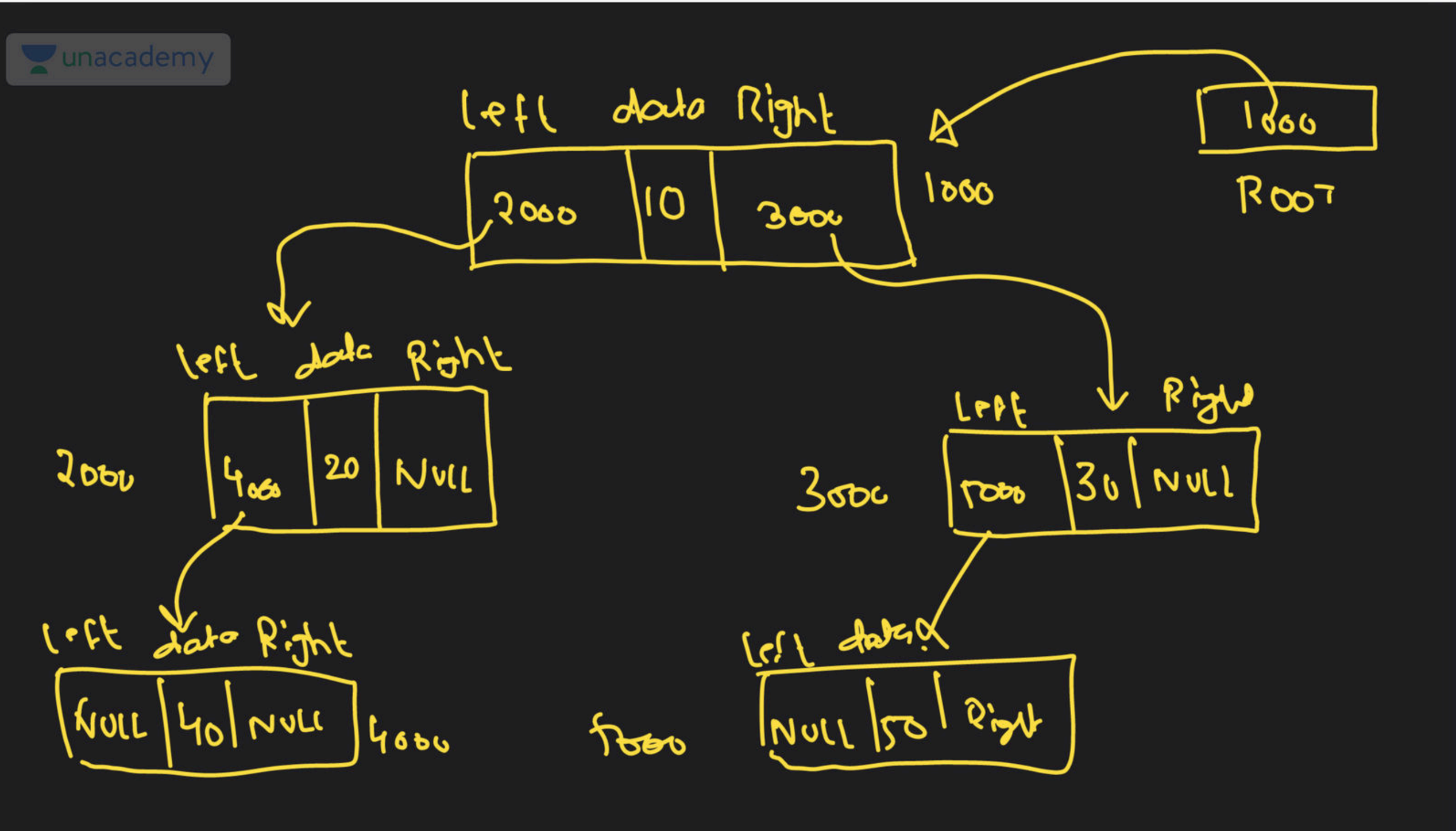
level	# Nodes
0	1
1	1
2	1
	3


Nodes
level
0
1
2
3
h
$n_{min} = h+1$

```
struct node {
    struct node *Left;
    int data;
    struct node *Right;
};
```






unacademy
left data Right
2000
10
3000
1000
1000
ROOT
left data Right
2000
4000
20
NULL
left data Right
3000
1000
30
NULL
left data Right
NULL
40
NULL
4000
5000
NULL
50
Right

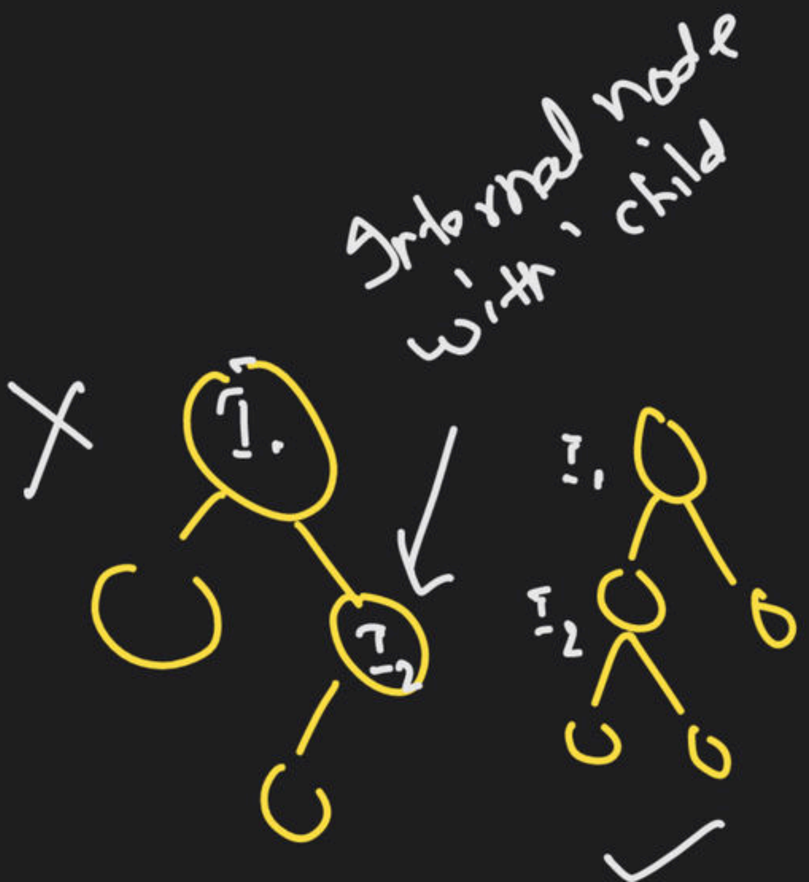
ROOT

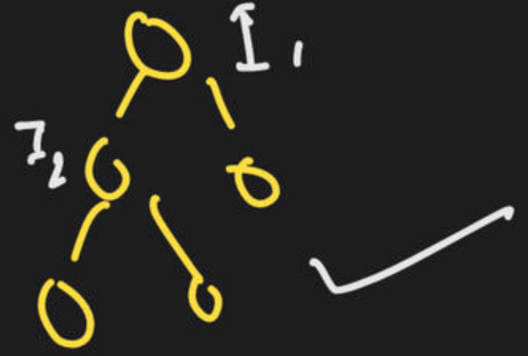
NULL

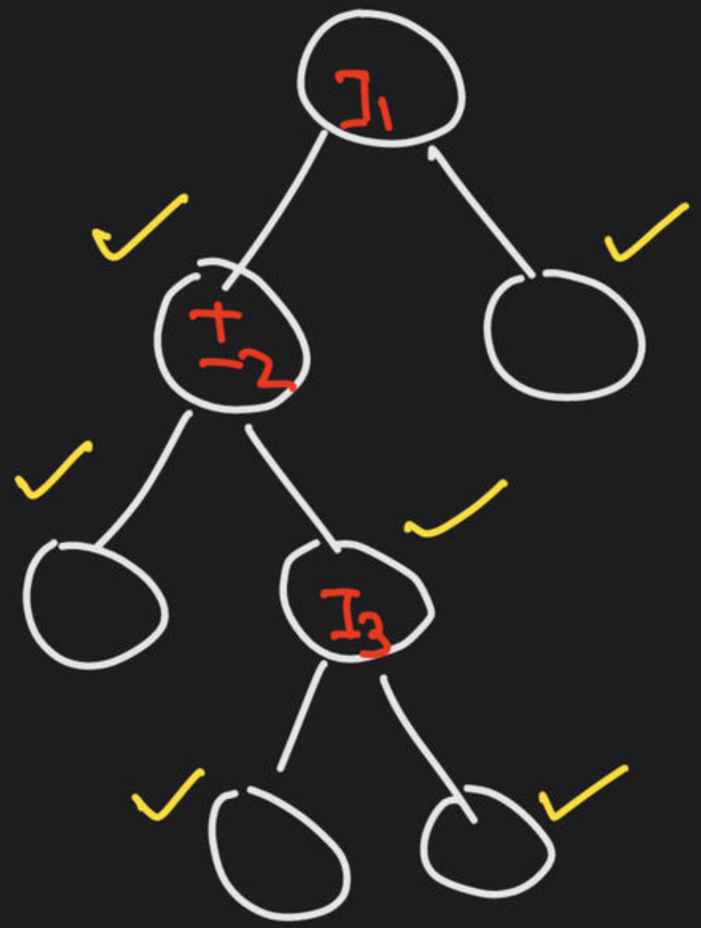
2-ary Tree : Binary tree in which every node has either 0 child or 2 child

OR

Binary tree in which every internal node has exactly 2 node







3 internal nodes

3 nodes of degree 2

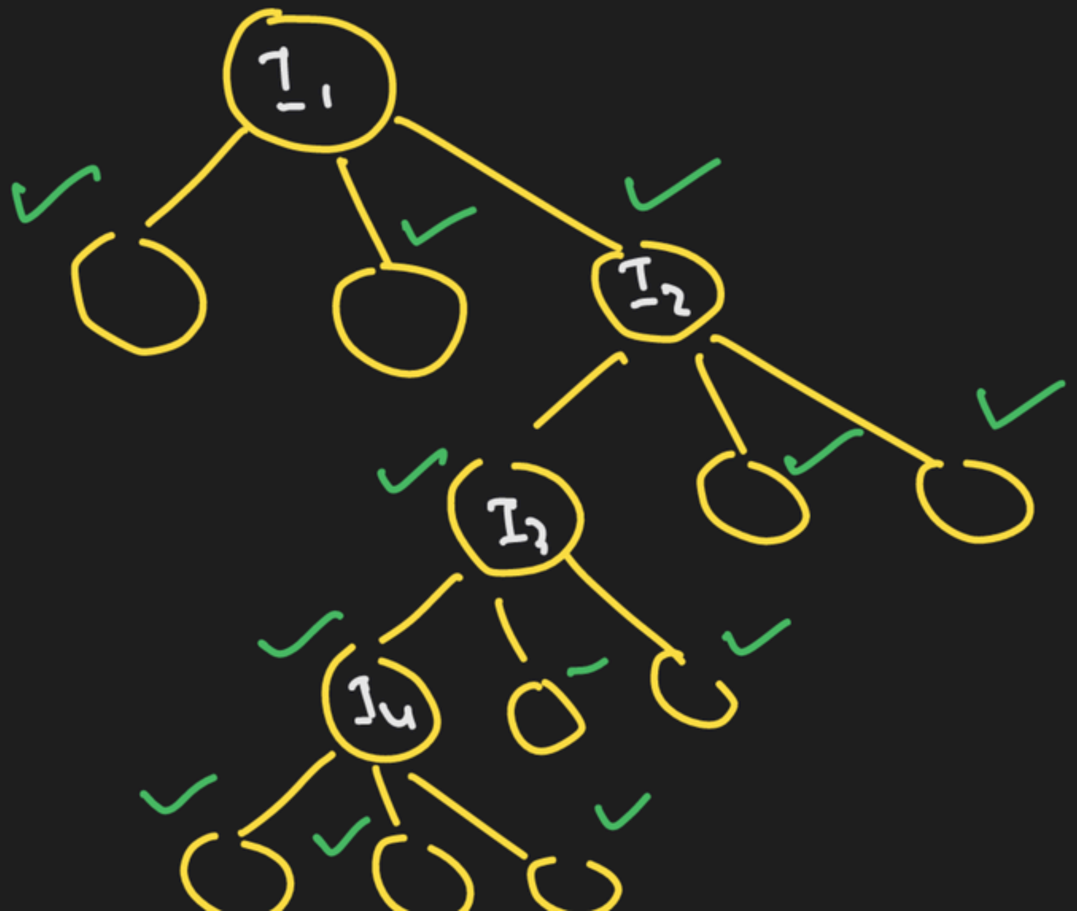
3 nodes with 2 child

$\Rightarrow$ 3×2

Total nodes = 3×2 + 1 (root)

unacademy

3-ary tree : A tree in which every internal node has exactly 3 childs



4 internal nodes ⇒ 3 childs

4 x 3

No. of internal nodes

Every 'internal' node has 3 child

Total nodes : $4 \times 3 + 1$ (root)

k-ary tree : Every internal node has exactly k children

let I : no. of internal nodes

Total nodes : $I \times k + 1$

$$N = k \cdot I + 1$$

unacademy

$$N = k \cdot I + 1$$

Don't learn any of these

# of leaf nodes + # of internal nodes : $kI + 1$

$$L + I = k \cdot I + 1$$

$$L = kI - I + 1$$

$$\boxed{L : I(k-1) + 1}$$

1. $N = kI + 1$

2. $L = (k-1)I + 1$

$$N = f(L)$$

$$(L-1) = (k-1)\cdot I$$

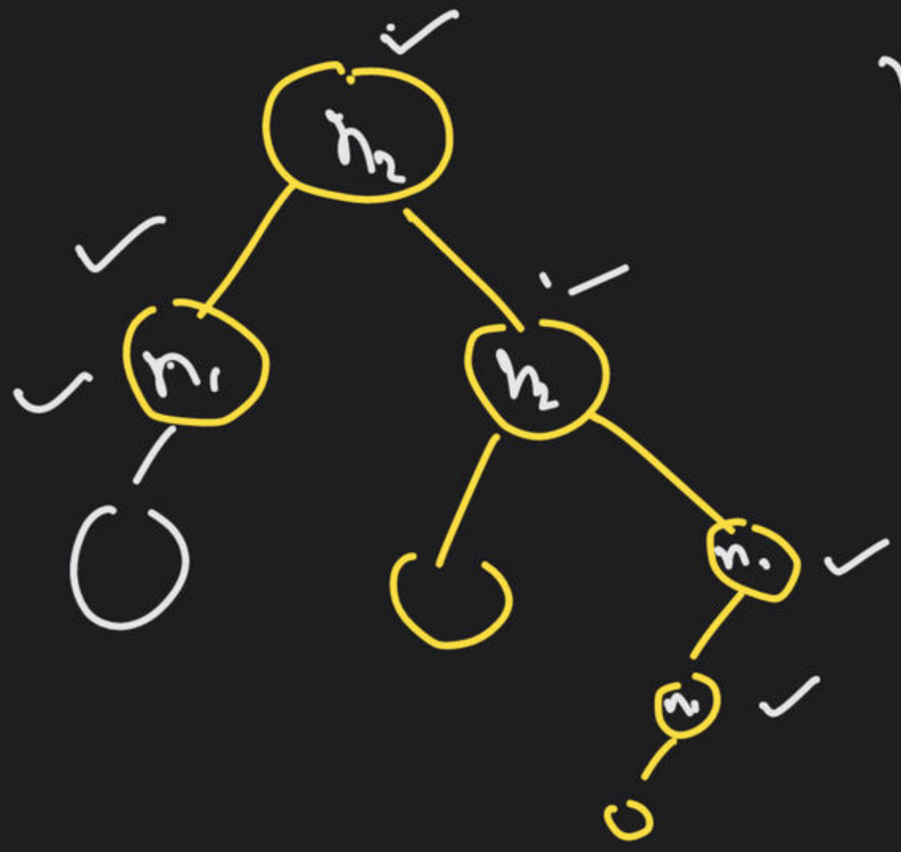
$$I = \frac{(L-1)}{(k-1)}$$

$$N = k\left(\frac{L-1}{k-1}\right) + 1$$

$$= \frac{kL - k + k - 1}{k-1}$$

$$N = \frac{kL-1}{k-1}$$

Q) A binary tree has 6 nodes of degree 1, 12 nodes of degree 2. find the no. of leaf nodes.



$n_2 = 2 \Rightarrow 2 \times 2$

$n_1 = 3 \Rightarrow 3 \times 1$

$7 + 1$

6 nodes of degree 1 $\Rightarrow$ $6 \times 1 = 6$

12 nodes " " 2 $\Rightarrow$ $12 \times 2 = 24$

Total no. of nodes: $6 \times 1 + 12 \times 2 + \underset{\text{root}}{1}$

$$N = 31$$

#node with 0 degree + # nodes with 1 degree

+ # nodes with 2 degree = 31

unacademy

nodes with 0 degree + # nodes with 1 degree + # nodes with 2 degree = 31

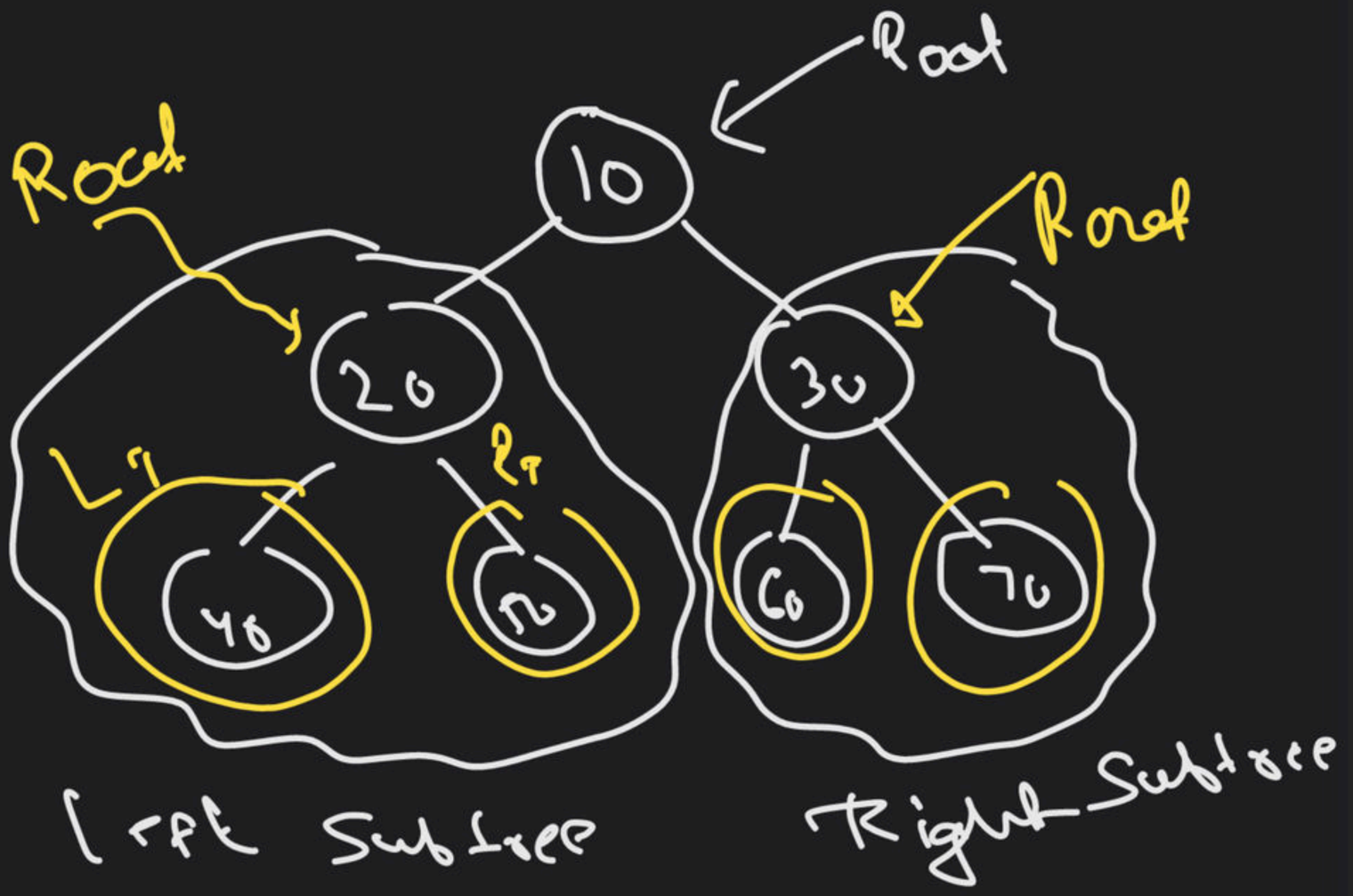
$$L + 6 + 12 = 31$$

$$L + 18 = 31$$

$$\boxed{L = 13}$$

unacademy

Tree Traversal

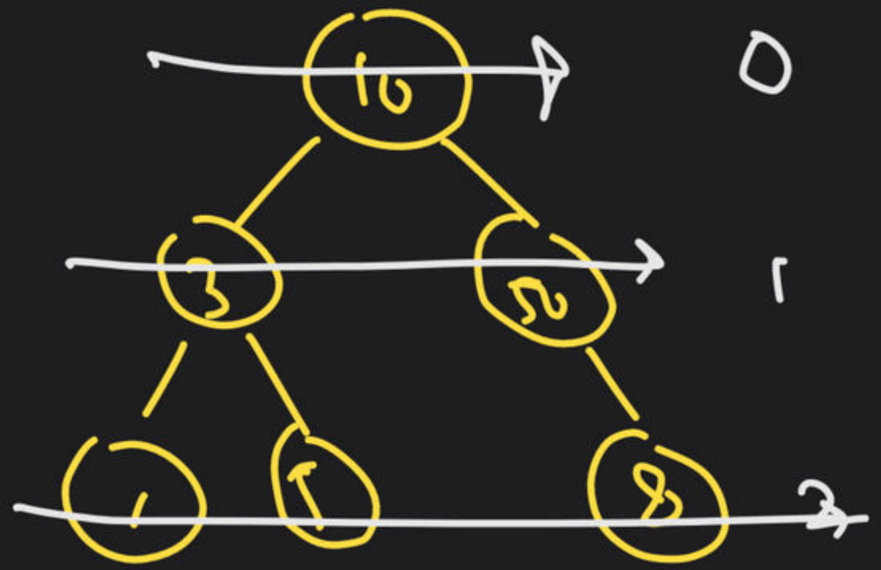


Root, L_T, R_T

unacademy

Tree Traversal

Level order traversal



0 10

1

2

10, 3, 12, 1, 7, 8

Depth order

Root, L_T, R_T

$3! = 6$ ways

1) $L_T, R_T, Root$
2) $L_T, Root, R_T$
3) $Root, L_T, R_T$
4) $R_T, L_T, Root$
5) $R_T, Root, L_T$
6) $Root, R_T, L_T$

L_T is traversed before R_T

1) Root, L_T, R_T — Preorder Traversal

2) L_T, Root, R_T — Inorder Traversal

3) L_T, R_T, Root — Postorder Traversal

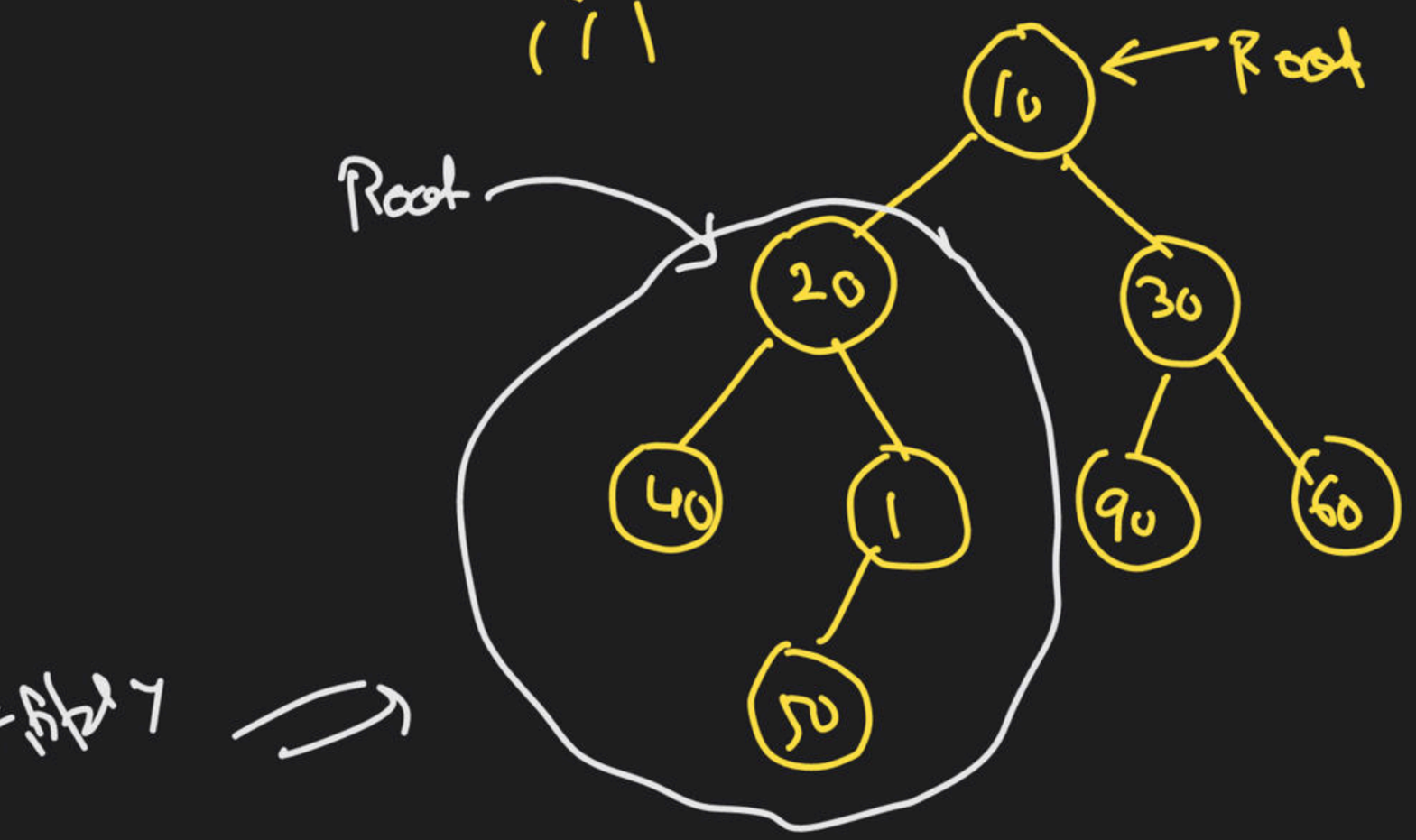
Pre-order Traversal

1.) Visit/Print/Process the Root node.
2) Traverse the left subtree (L_T) in Preorder
3) Traverse the right subtree (R_T) in Preorder

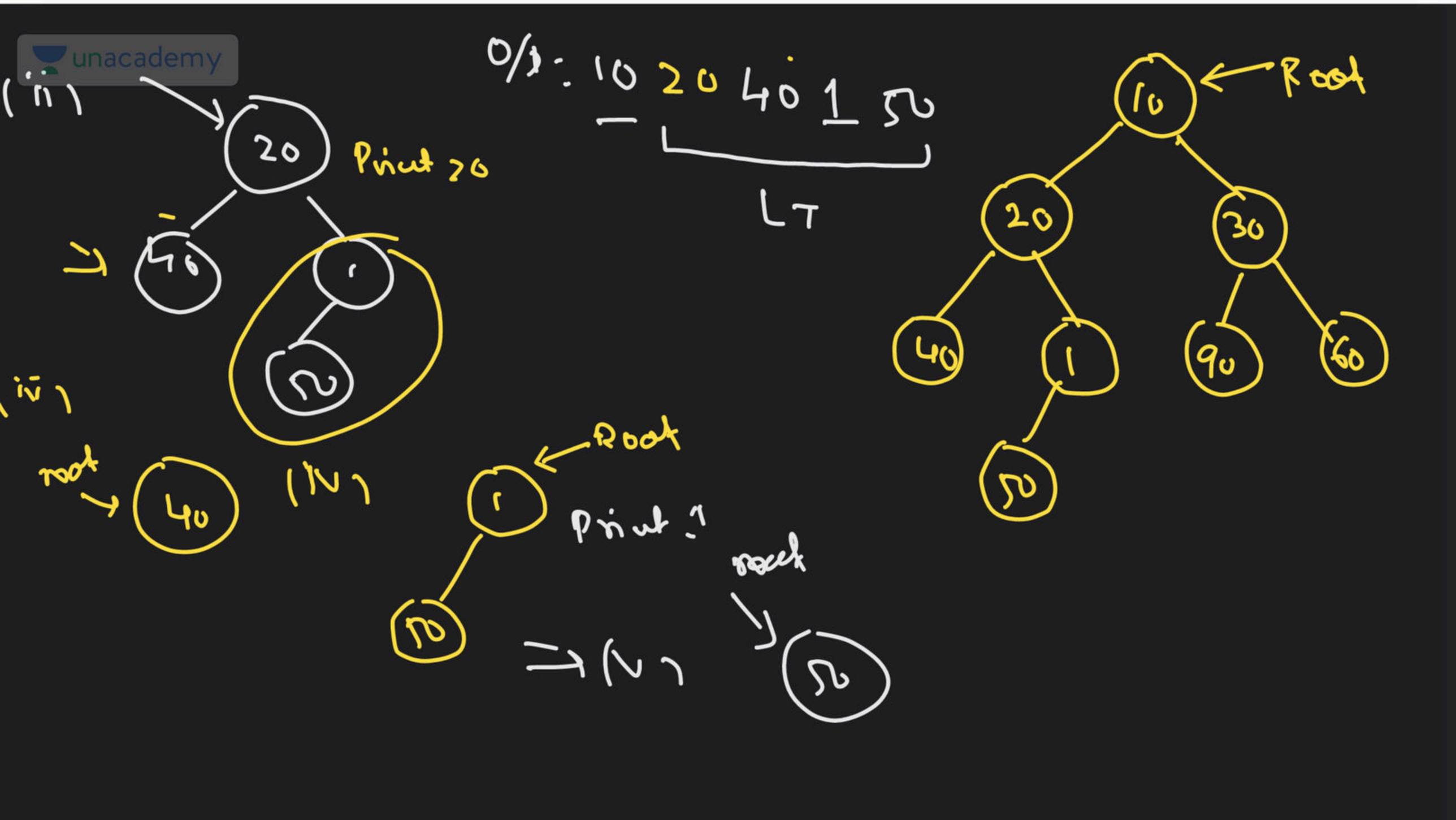
unacademy

Print = 10

O/P: 10

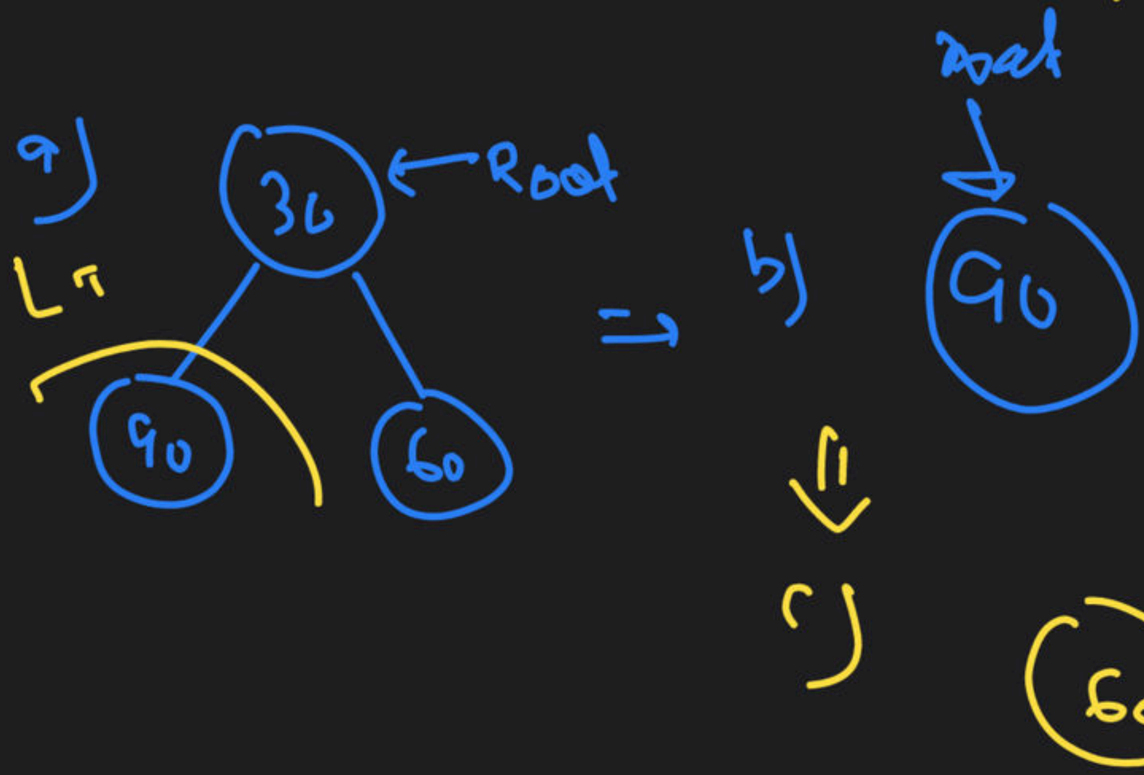


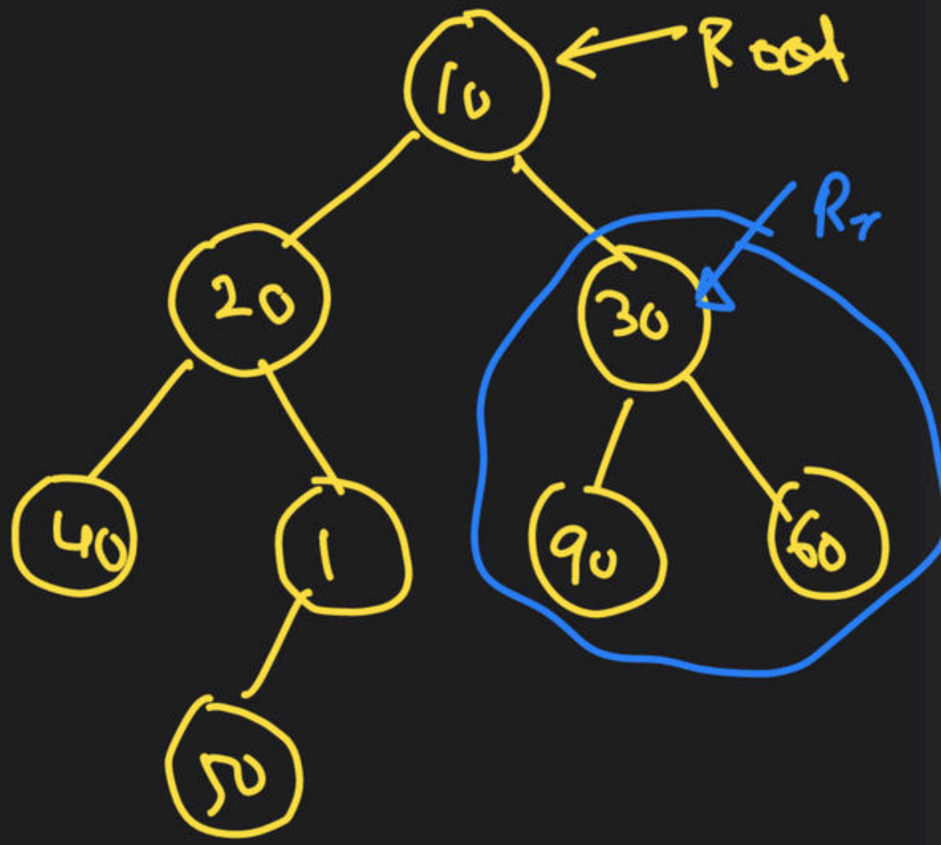
Algo

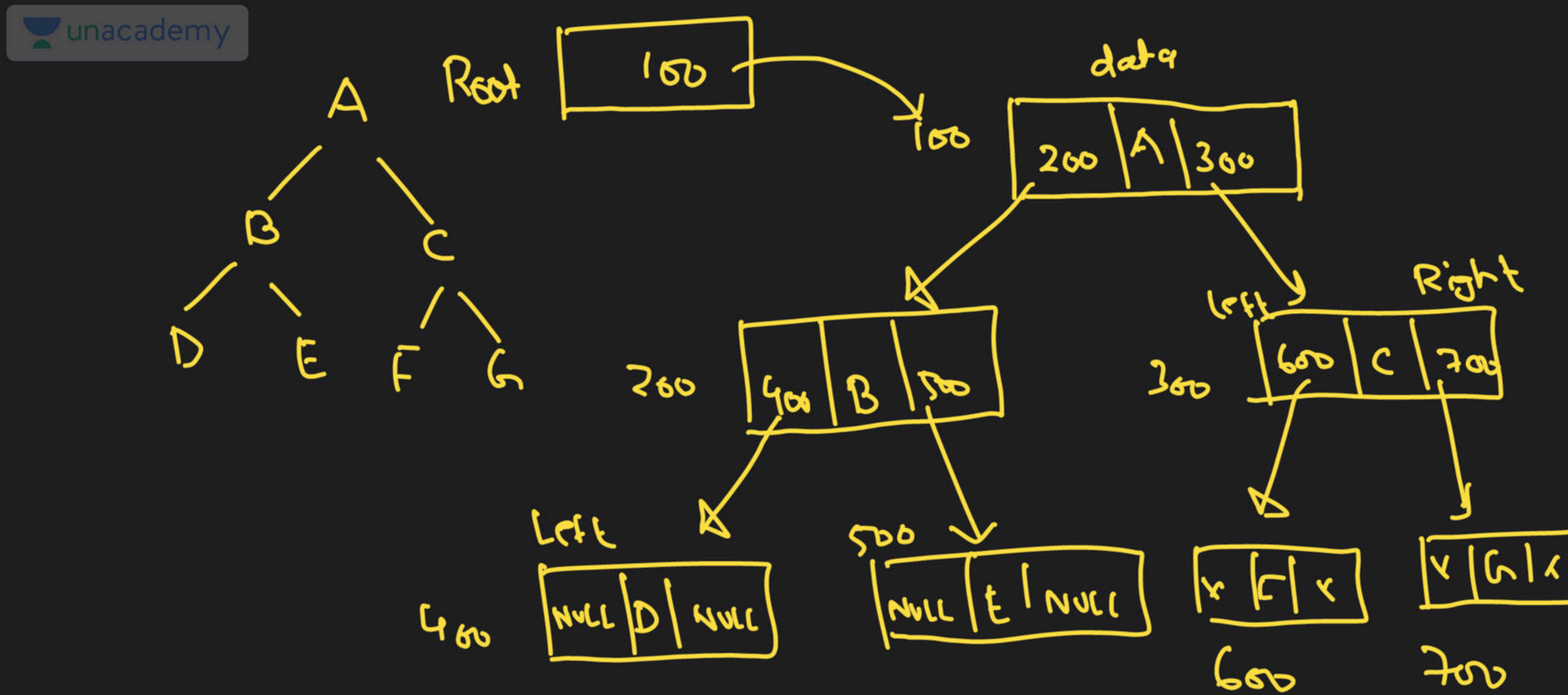

O/P: 10 20 40 1 50
LT
Root
10
20
30
40
1
90
60
50
(i)
20
Print 20
40
1
2
(ii)
root
40
(iii)
Root
1
Print 1
2
root
2
(iv)

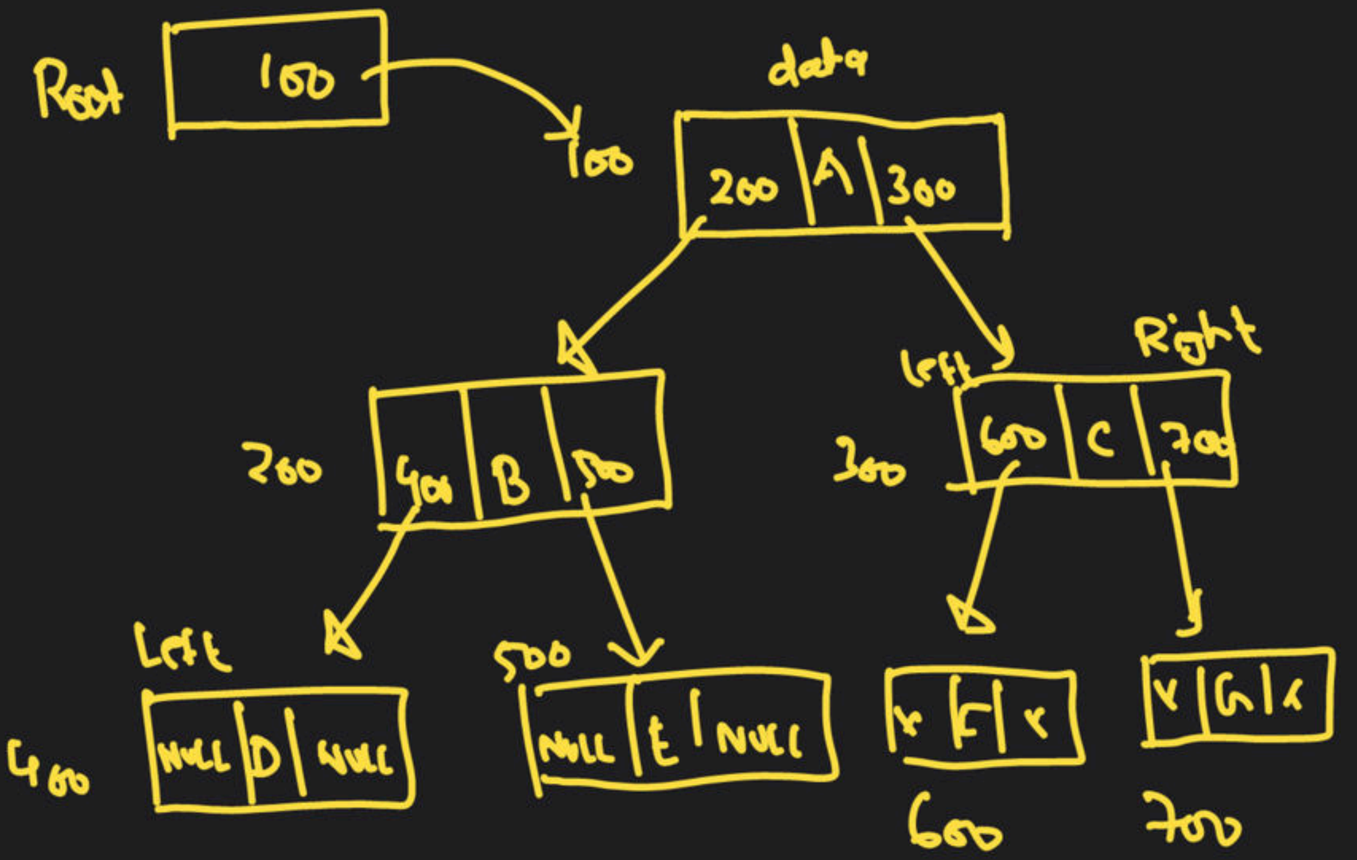
0/1 - 10 20 40 1 50 30 90 60

Root L R T






unacademy
A
B
C
D
E
F
G
Root
100
100
data
200
A
300
200
400
B
500
Left
Right
300
600
C
700
Left
400
NULL
D
NULL
500
NULL
E
NULL
x
F
x
x
G
x
600
700



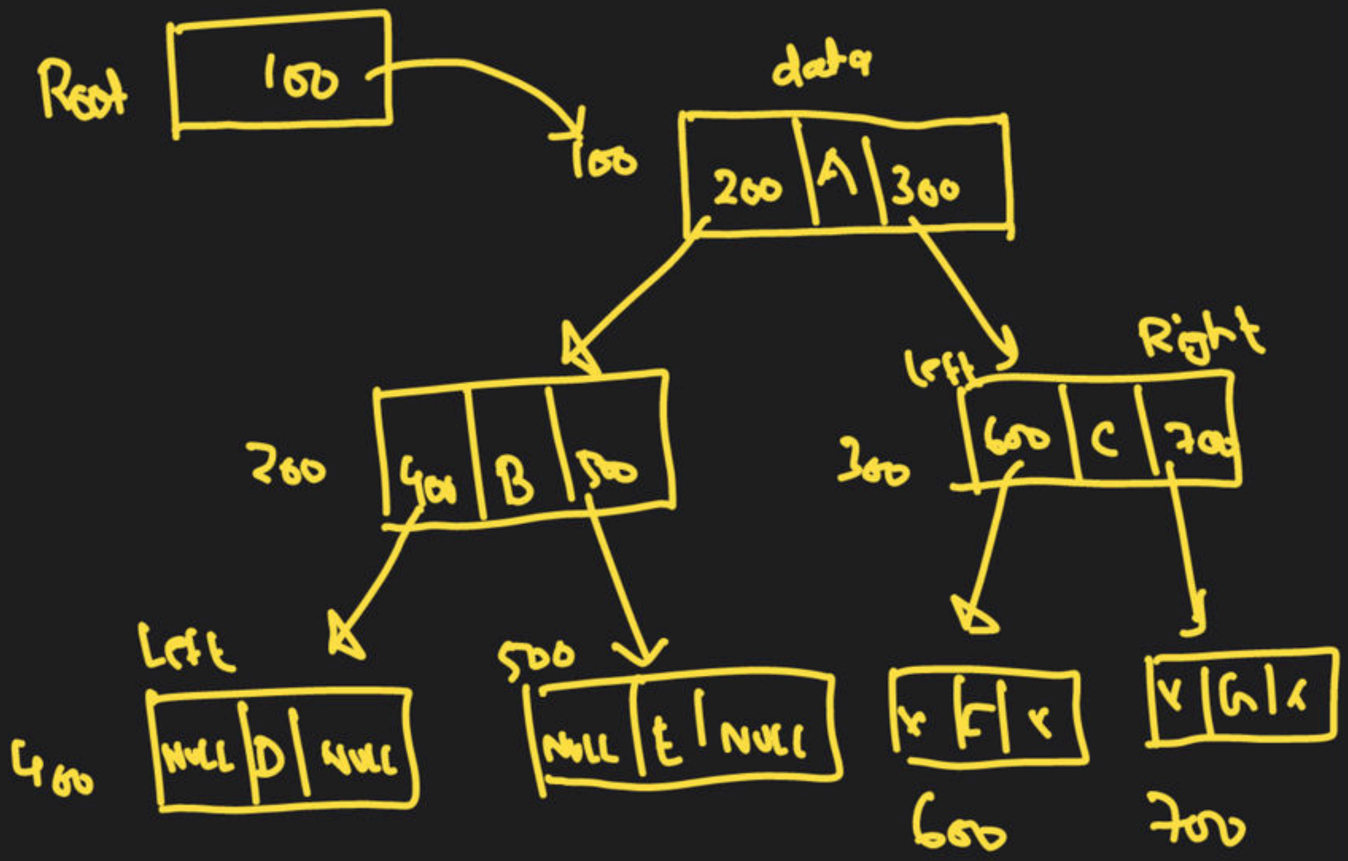
Ptr → 100

Ptr → Left

Ptr → data

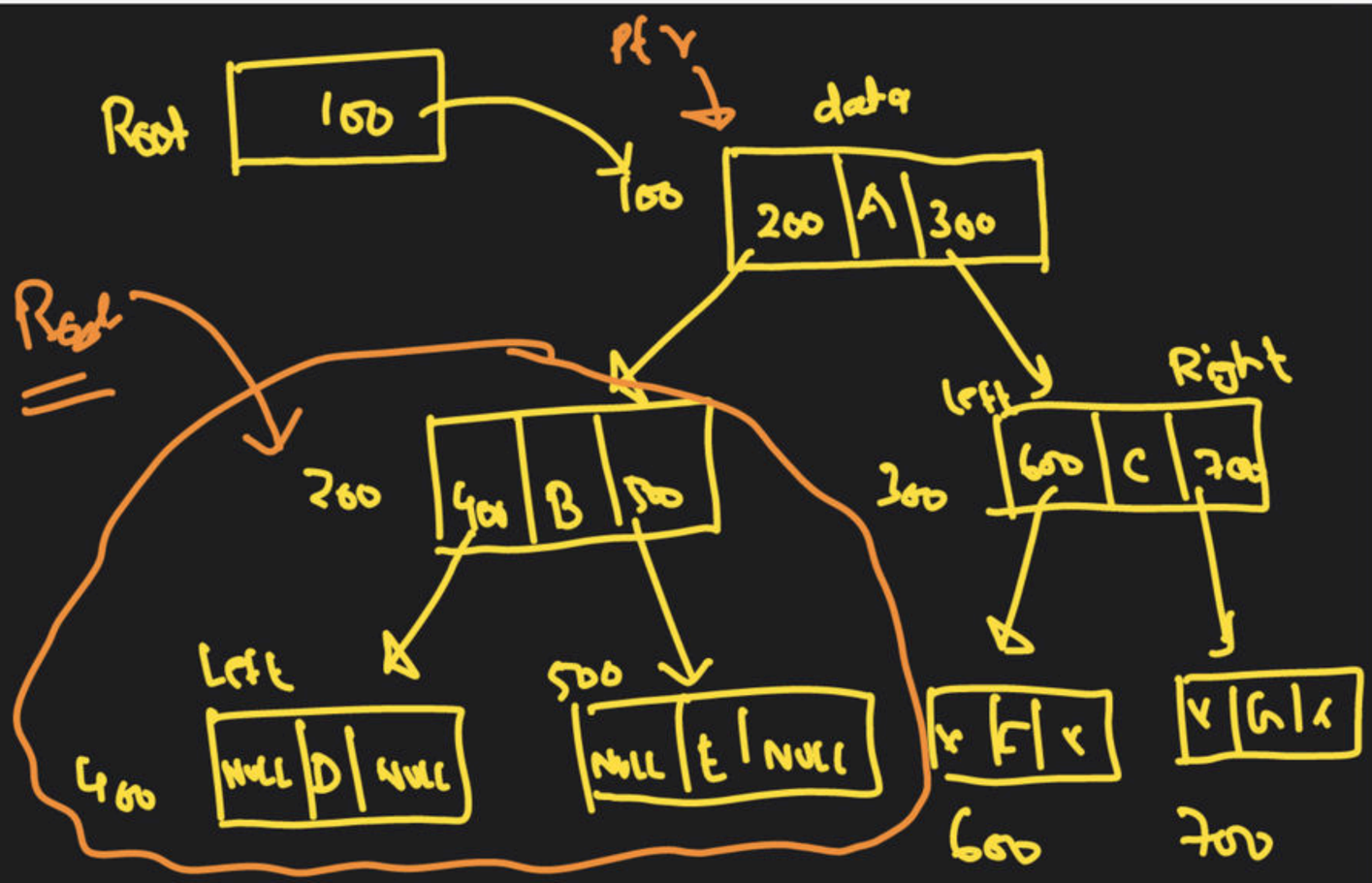
Ptr → right

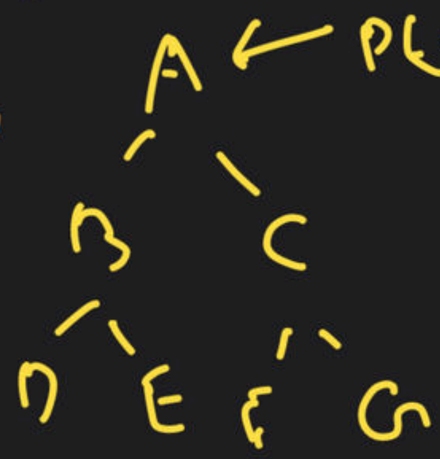
```
void Preorder(struct Node *Ptr)
{
    if (Ptr == NULL)
        return;

}
```



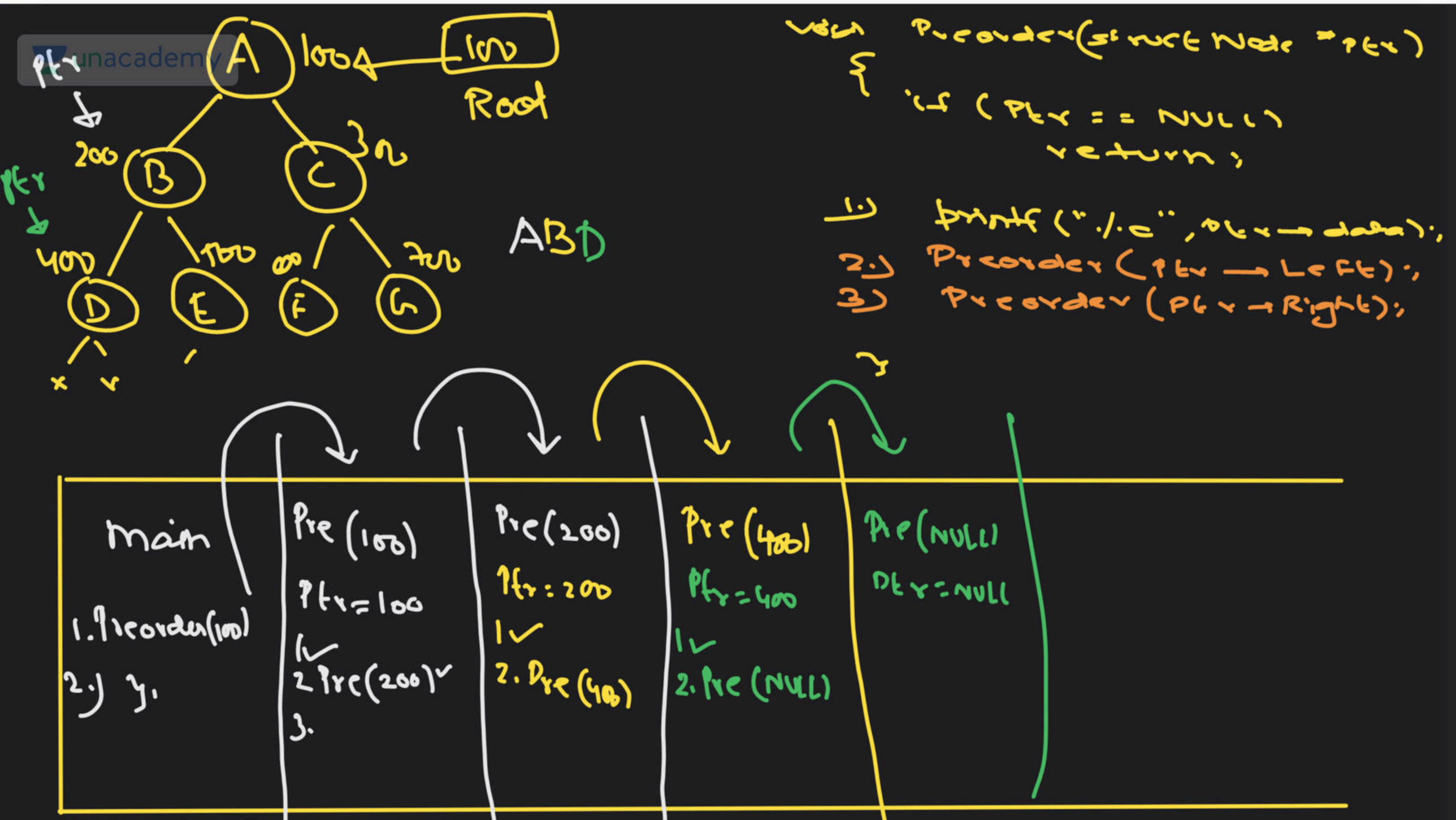

```
void main(){
    ...
    Preorder(Root);
```

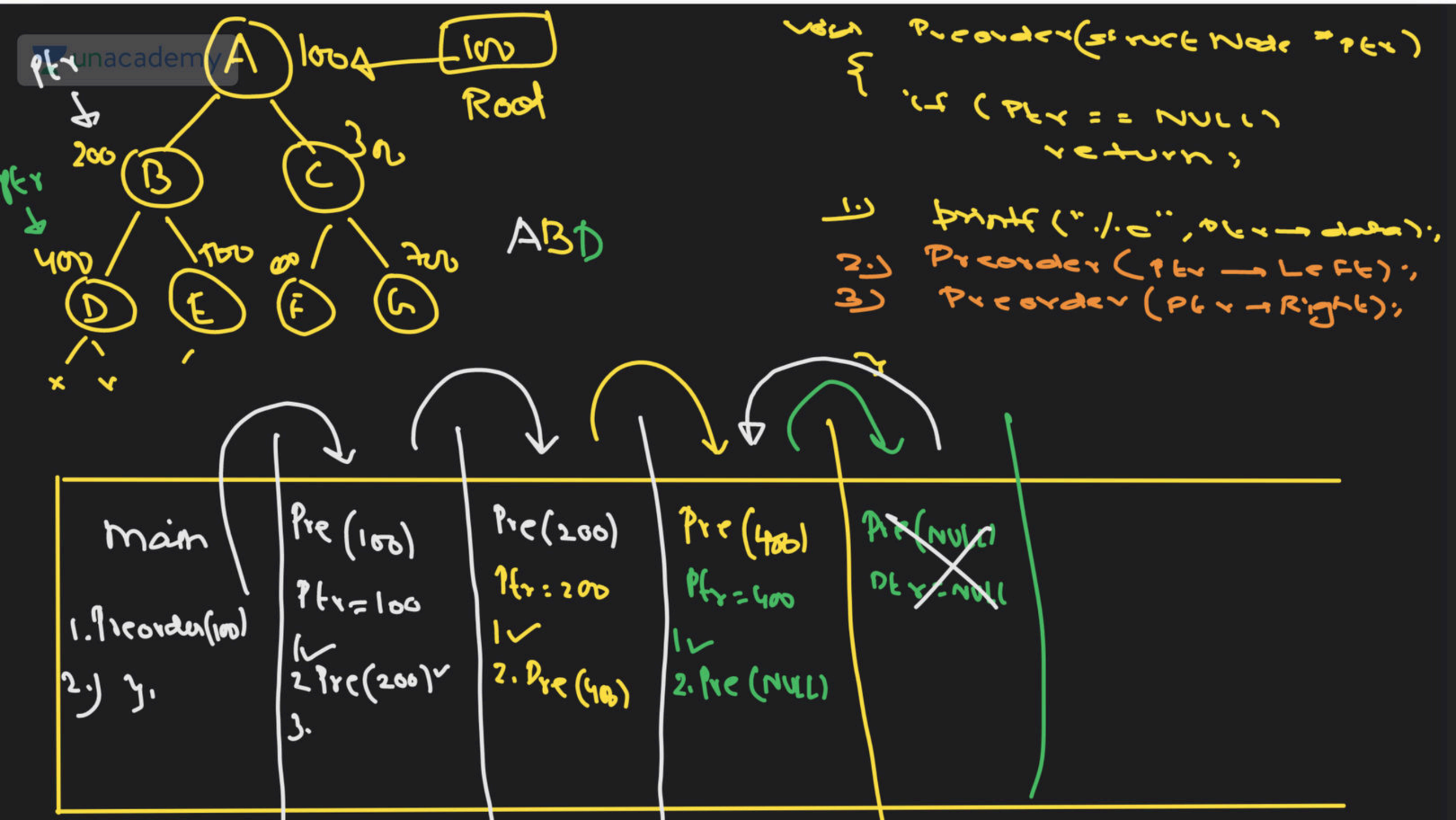
```
void Preorder(struct Node *Ptr)
{
    if (Ptr == NULL)
        return;

1.) printf("%c", Ptr->data);
2.) Preorder(Ptr->Left);
3.) Preorder(Ptr->Right);

}
```

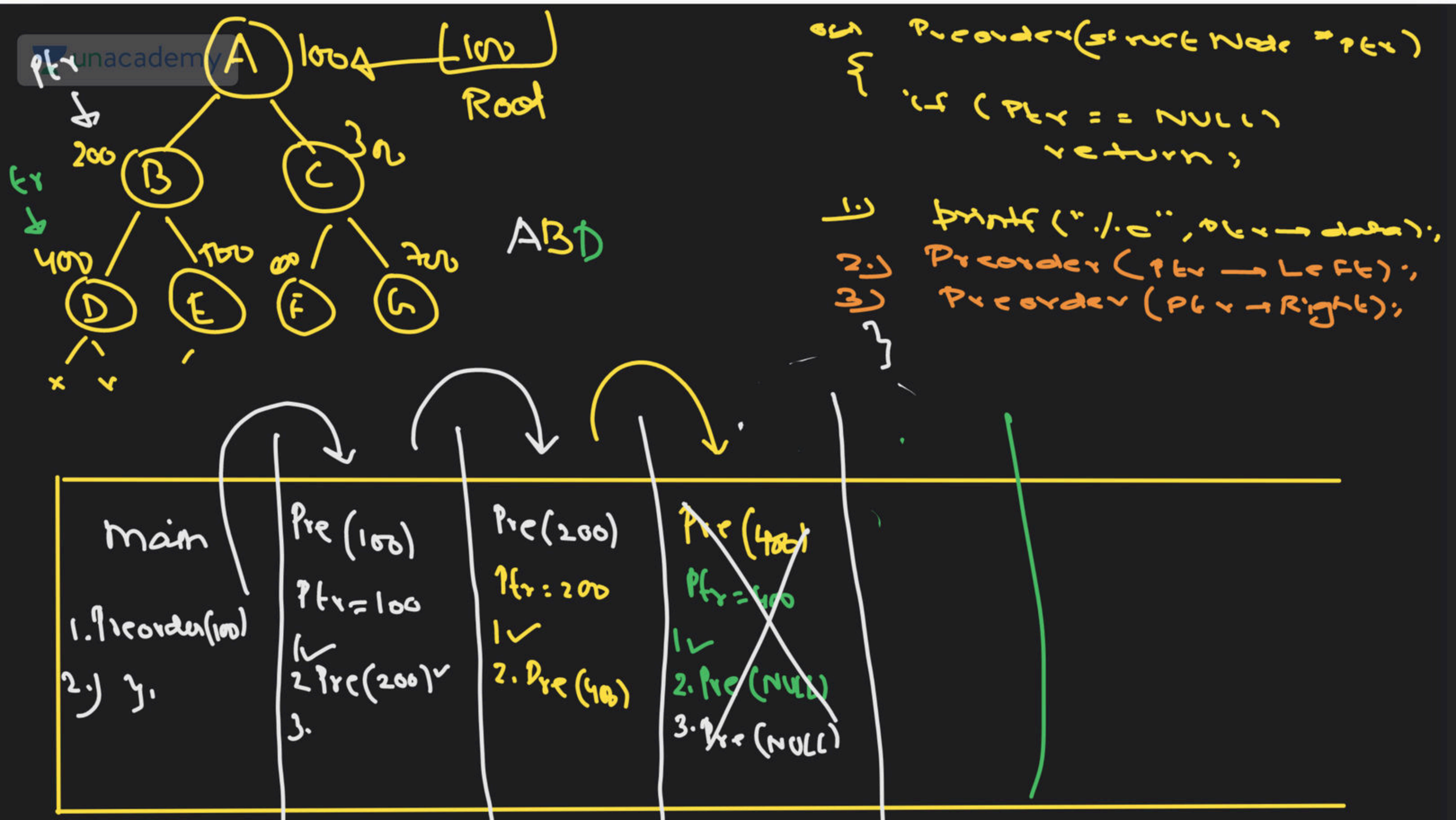


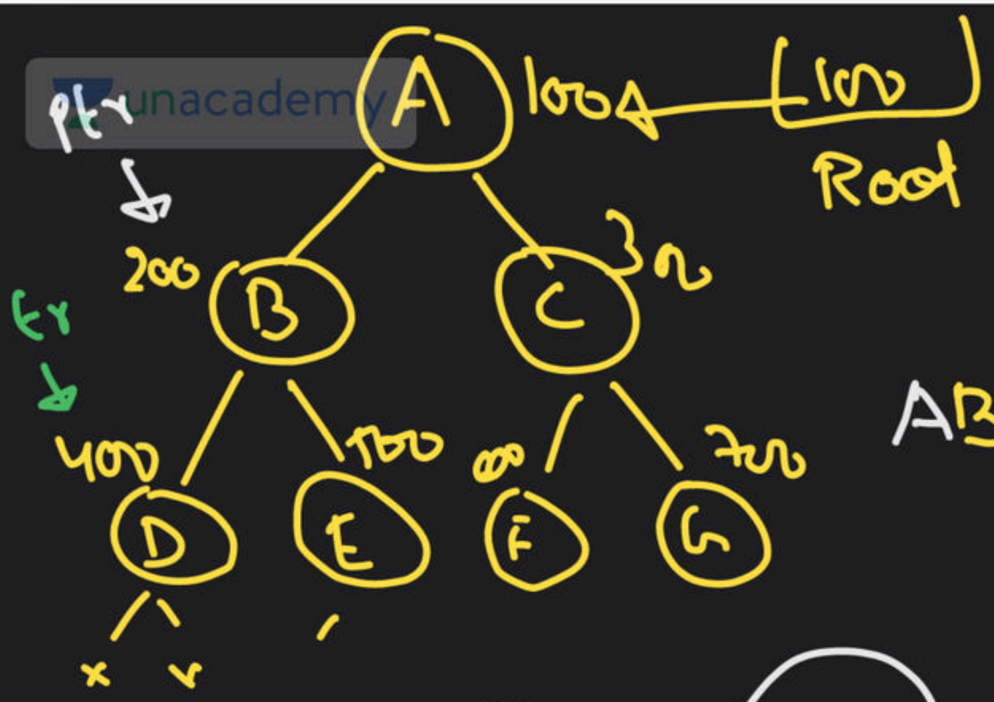



```
void main(){
    ...
    Preorder(Root);
}
```


ptr
A
1000
100
Root
200
B
C
ptr
400
D
E
F
G
ABD
void Preorder(struct Node *ptr)
{
if (ptr == NULL)
return;
1.) printf("%d", ptr->data);
2.) Preorder(ptr->Left);
3.) Preorder(ptr->Right);
}
main
1. Preorder(100)
2.) y.
Pre(100)
ptr=100
1✓
2 Pre(200)✓
3.
Pre(200)
ptr:200
1✓
2. Pre(400)
Pre(400)
ptr=400
1✓
2. Pre(NULL)
Pre(NULL)
ptr=NULL

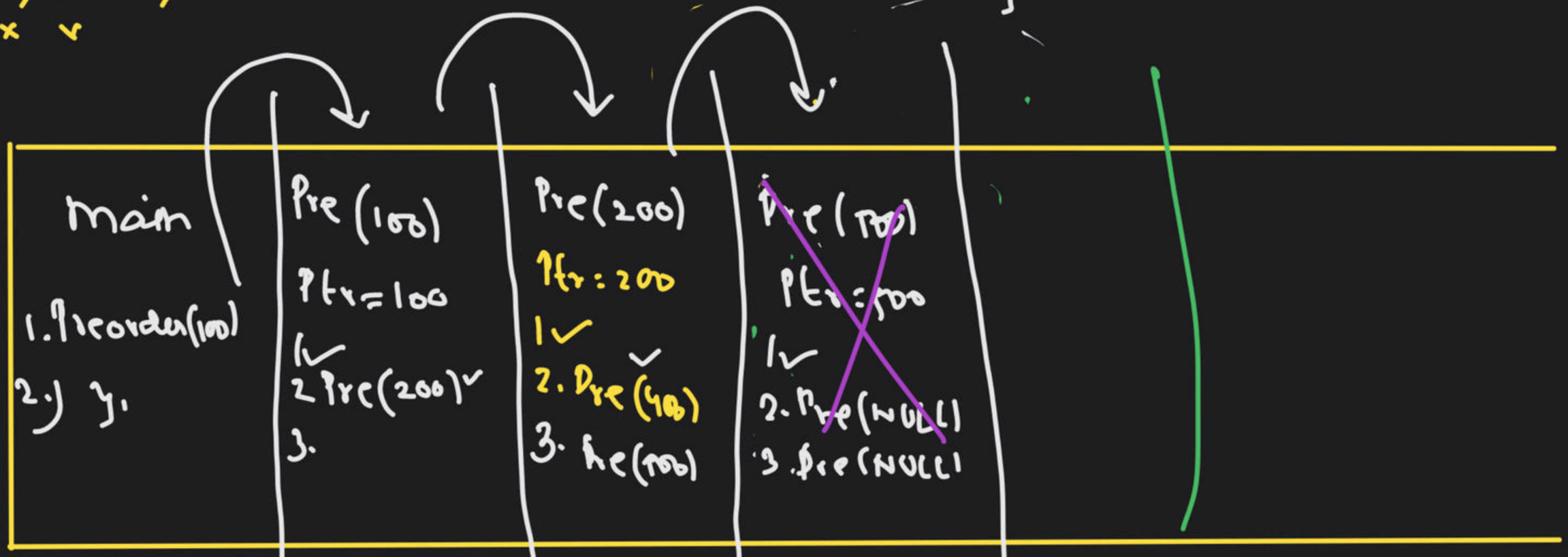

ptr
A
1000
100
Root
200
B
C
ptr
400
100
D
E
F
G
ABD
void Preorder(struct Node *ptr)
{
if (ptr == NULL)
return;
1.) printf("%d", ptr->data);
2.) Preorder(ptr->Left);
3.) Preorder(ptr->Right);
main
1. Preorder(100)
2.) y.
Pre(100)
Ptr=100
1✓
2 Pre(200)✓
3.
Pre(200)
Ptr:200
1✓
2. Pre(400)
Pre(400)
Ptr=400
1✓
2. Pre(NULL)
Ptr(NULL)
Ptr=NULL

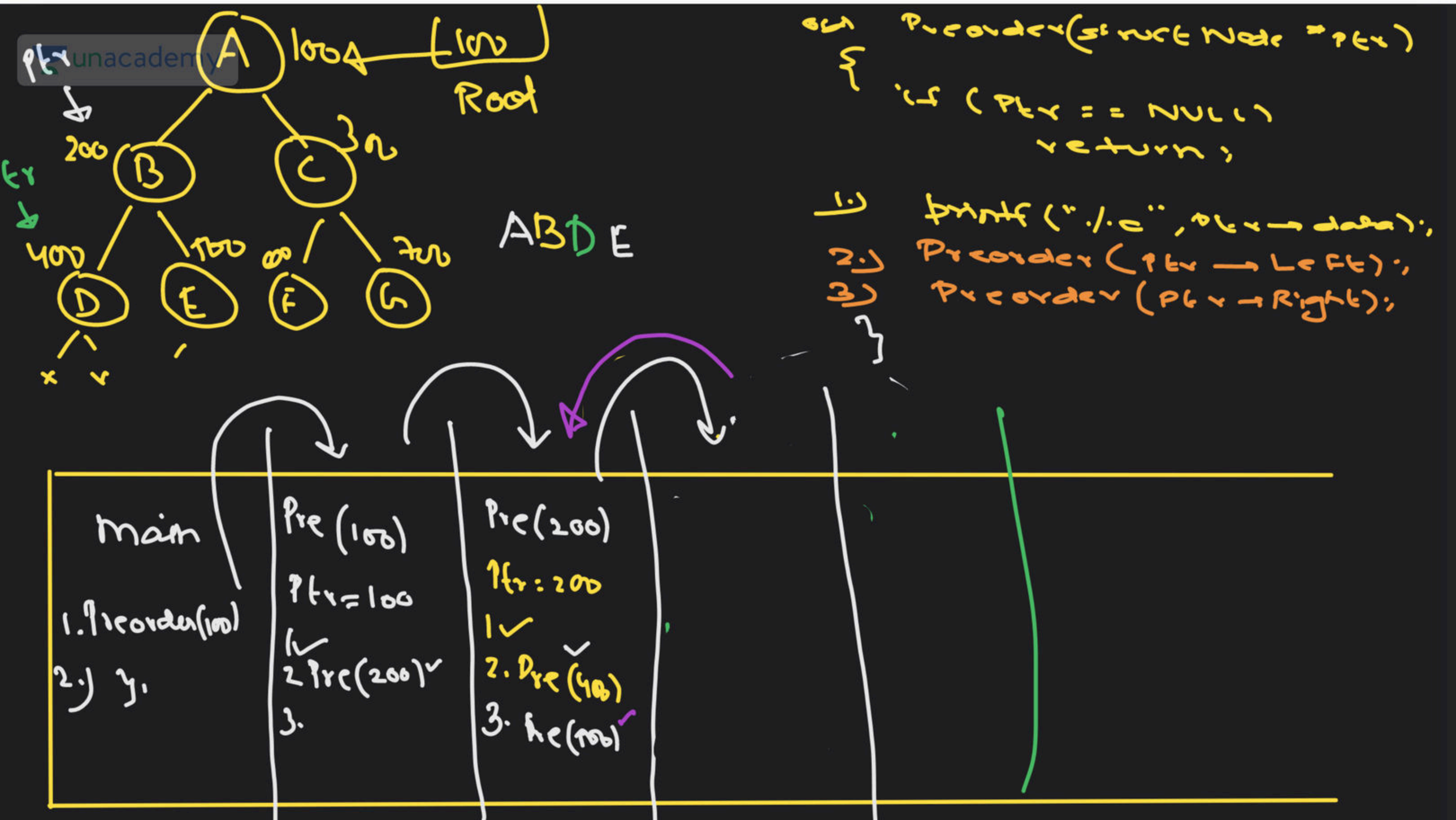

ptr
A
1000
100
Root
200
B
C
ptr
400
D
100
E
F
G
ABD
void Preorder(struct Node *ptr)
{
if (ptr == NULL)
return;
1.) printf("%c", ptr->data);
2.) Preorder(ptr->Left);
3.) Preorder(ptr->Right);
}
main
1. Preorder(100)
2.) y.
Pre(100)
ptr=100
1✓
2 Pre(200)✓
3.
Pre(200)
ptr:200
1✓
2. Pre(400)
Pre(400)
ptr=400
1✓
2. Pre(NULL)
3. Pre(NULL)

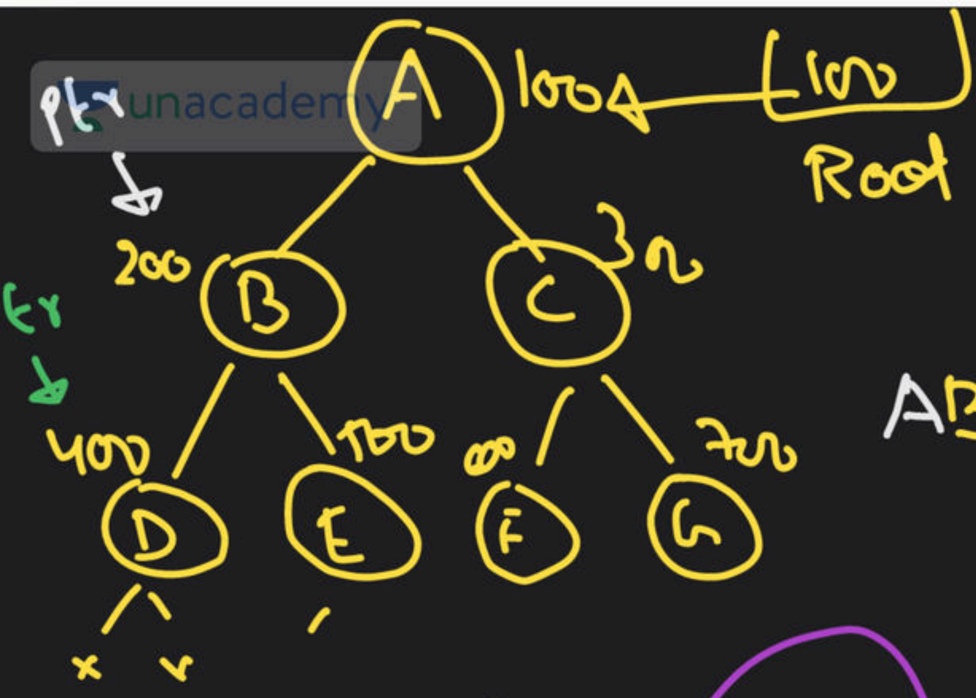


ABDE

```
void Preorder(struct Node *ptr)
{
  if (ptr == NULL)
      return;
1.)  printf("%d", ptr->data);
2.)  Preorder(ptr->Left);
3.)  Preorder(ptr->Right);
}
```

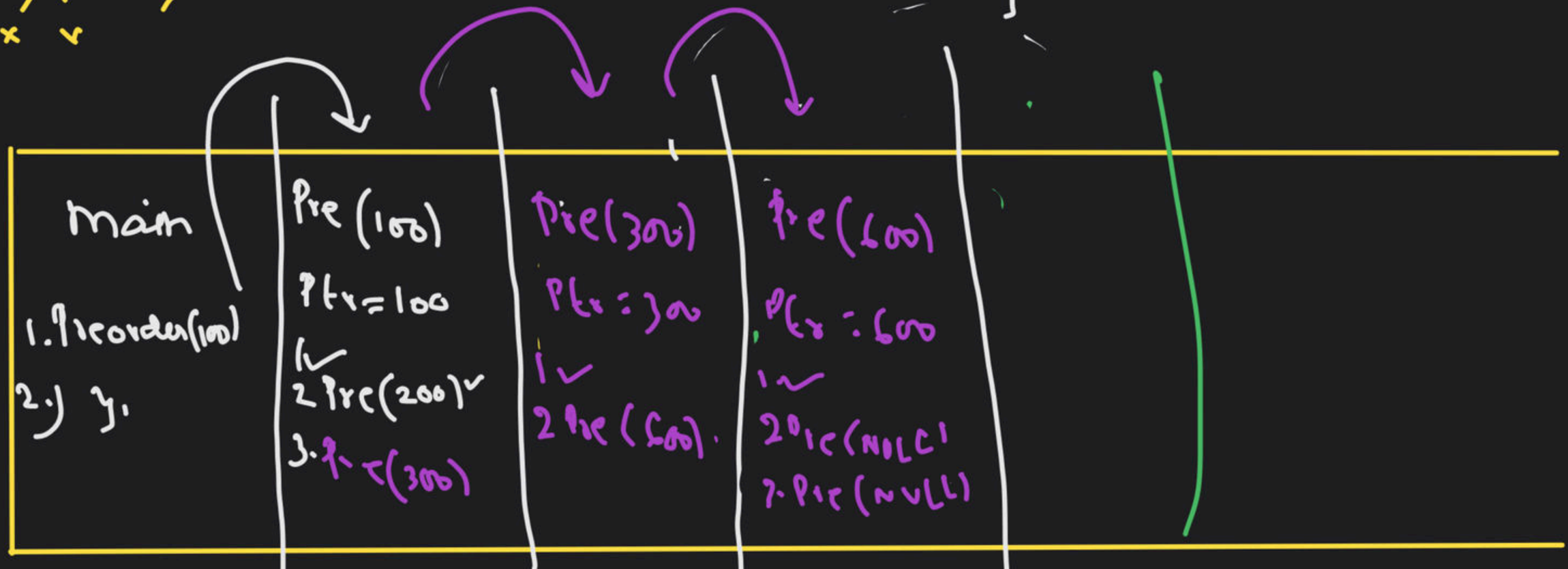


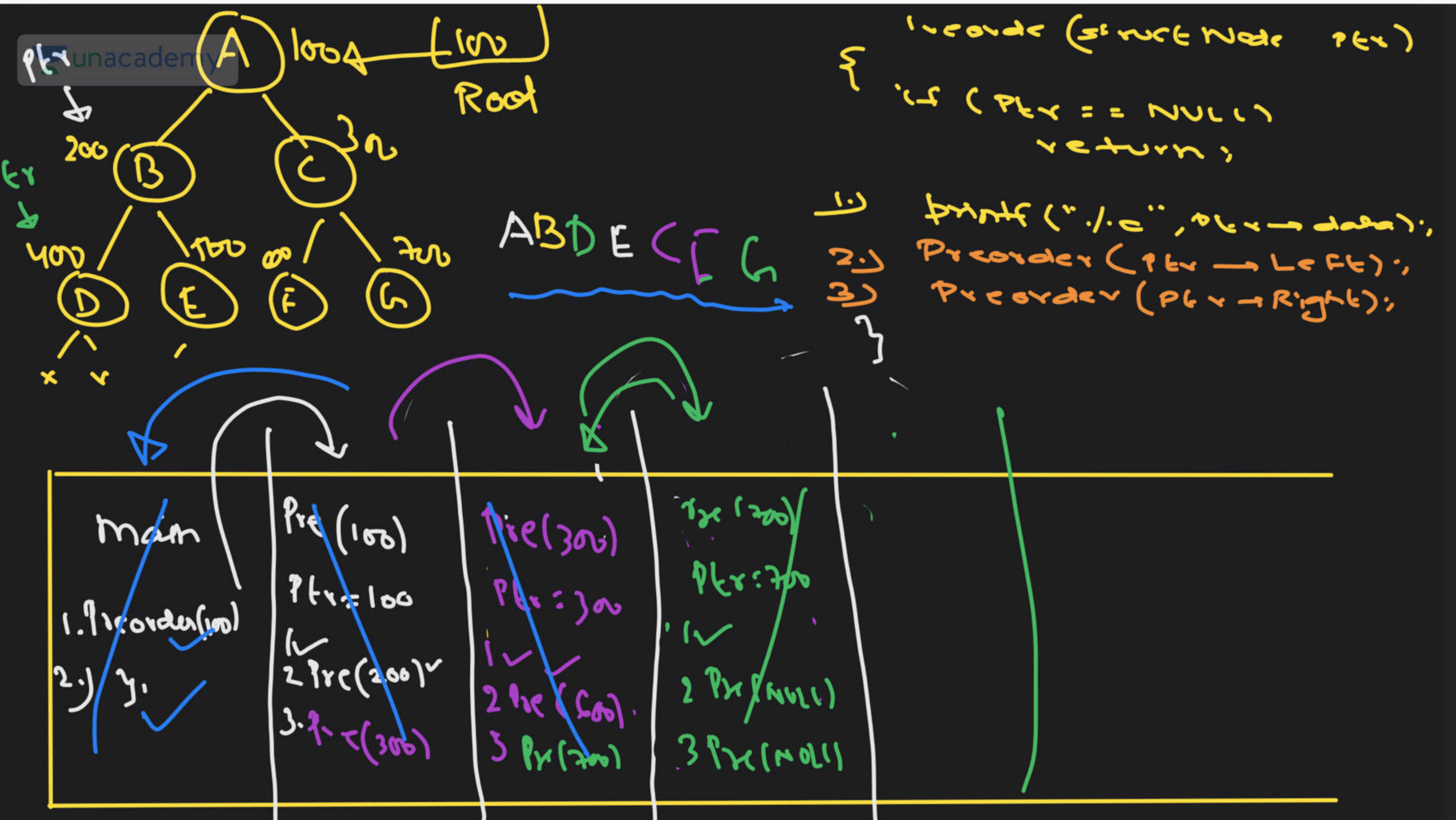

ptr
A
1004
100
Root
200
B
C
ptr
400
700
D
E
F
G
ABDE
void Preorder(struct Node *ptr)
{
if (ptr == NULL)
return;
1.) printf("%d", ptr->data);
2.) Preorder(ptr->Left);
3.) Preorder(ptr->Right);
}
main
1. Preorder(100)
2.) y.
Pre(100)
ptr=100
2 Pre(200)
3.
Pre(200)
ptr: 200
1
2. Pre(400)
3. Pre(700)



ABDECFG

```
Preorder (struct Node *ptr)
{
  if (ptr == NULL)
      return;
1.)  printf("%d", ptr->data);
2.)  Preorder(ptr->Left);
3.)  Preorder(ptr->Right);
}
```




unacademy
ptr
A
1004
100
Root
200
B
C
300
ptr
400
500
600
700
D
E
F
G
ABDECFG
Preorder (struct Node *ptr)
{
if (ptr == NULL)
return;
1.) printf("%c", ptr->data);
2.) Preorder(ptr->Left);
3.) Preorder(ptr->Right);
}
main
1. Preorder(100)
2.)
Pre(100)
ptr=100
2 Pre(200)
3. Pre(300)
Pre(300)
ptr=300
2 Pre(500)
3 Pre(700)
Pre(200)
ptr=200
2 Pre(NULL)
3 Pre(NULL)

P[x → NULL

unacademy

THANK YOU!

Here's to a cracking journey ahead!