unacademy

Linked List - Part VII

Course on Data Structure

Pankaj Sharma • Lesson 13 • July 23, 2024

CS & IT Engineering

Data Structure

Linked List

Lecture Number- 13

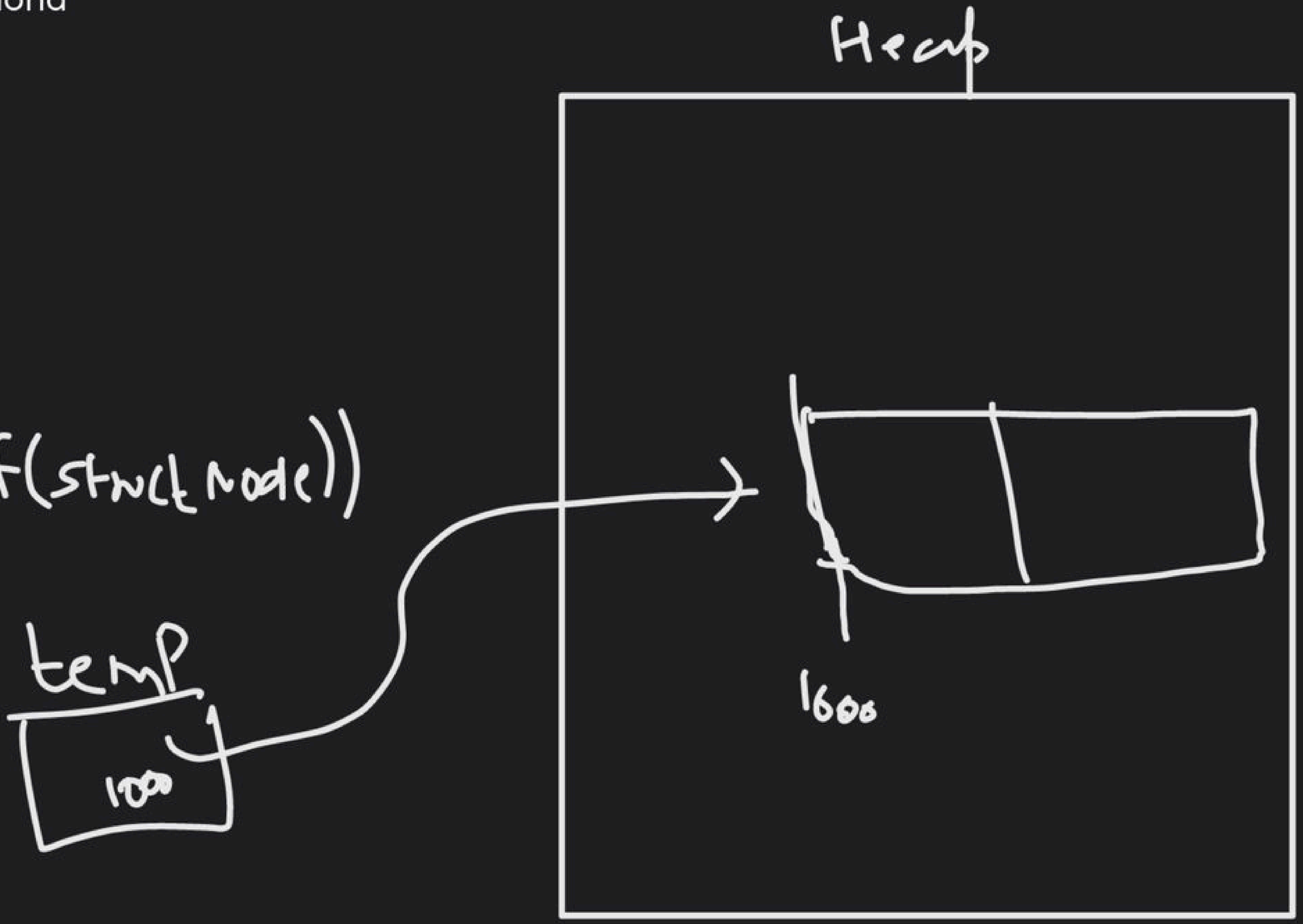
By- Pankaj Sir

▲ 1 • Asked by Adarsh

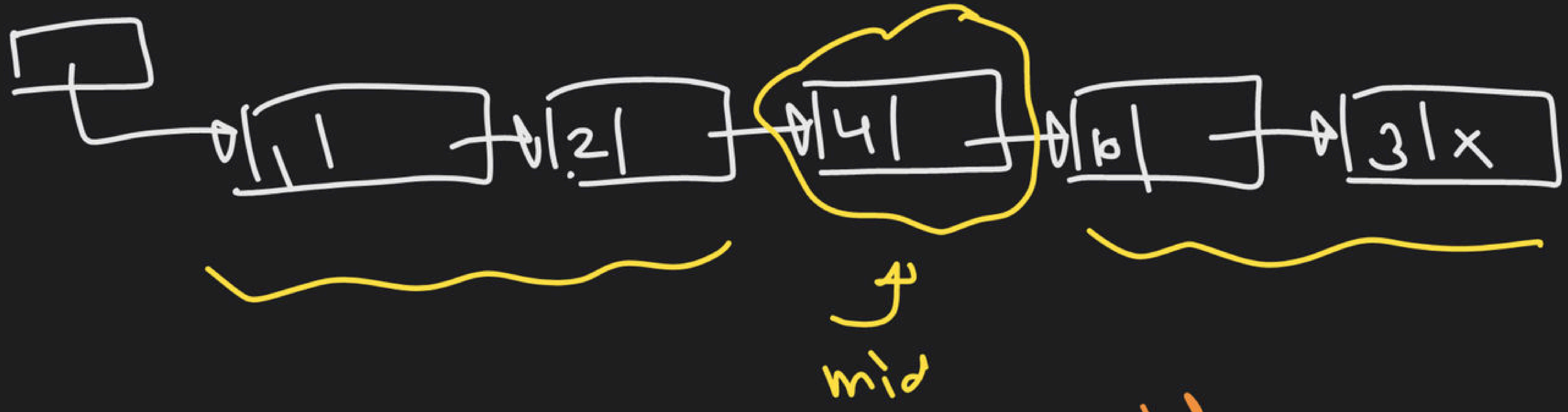
sir ek basic sa doubt hai plesse gussa mat hona

How to create a node :

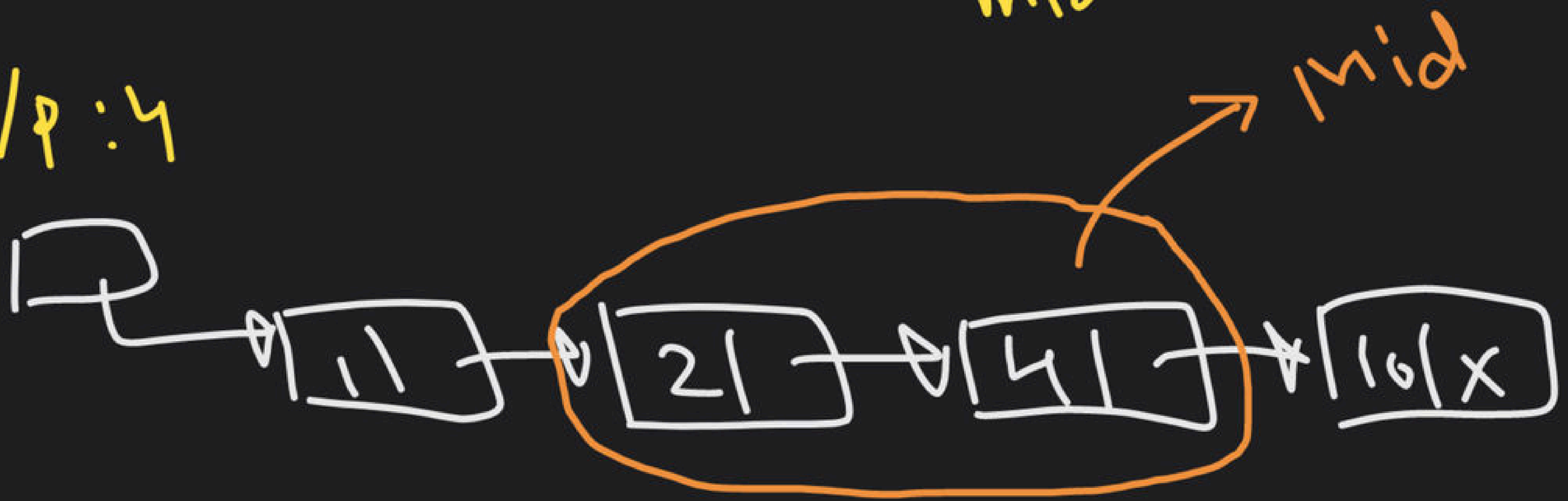
```
struct Node *temp;

temp = malloc(sizeof(struct Node))
```

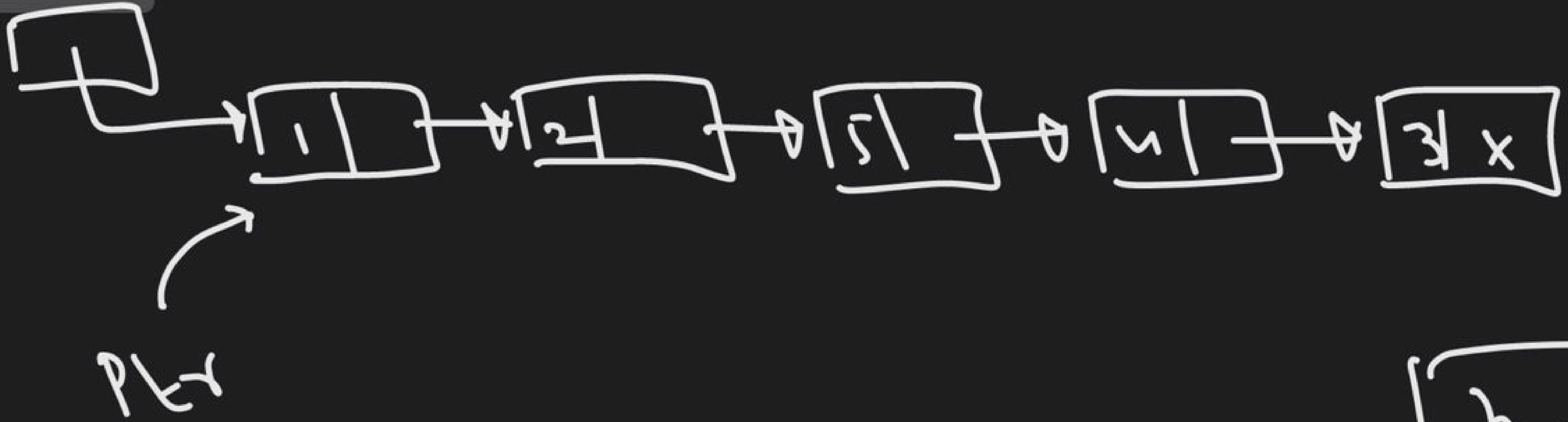


8. Given a LL, print mid node data.



o/p : 4

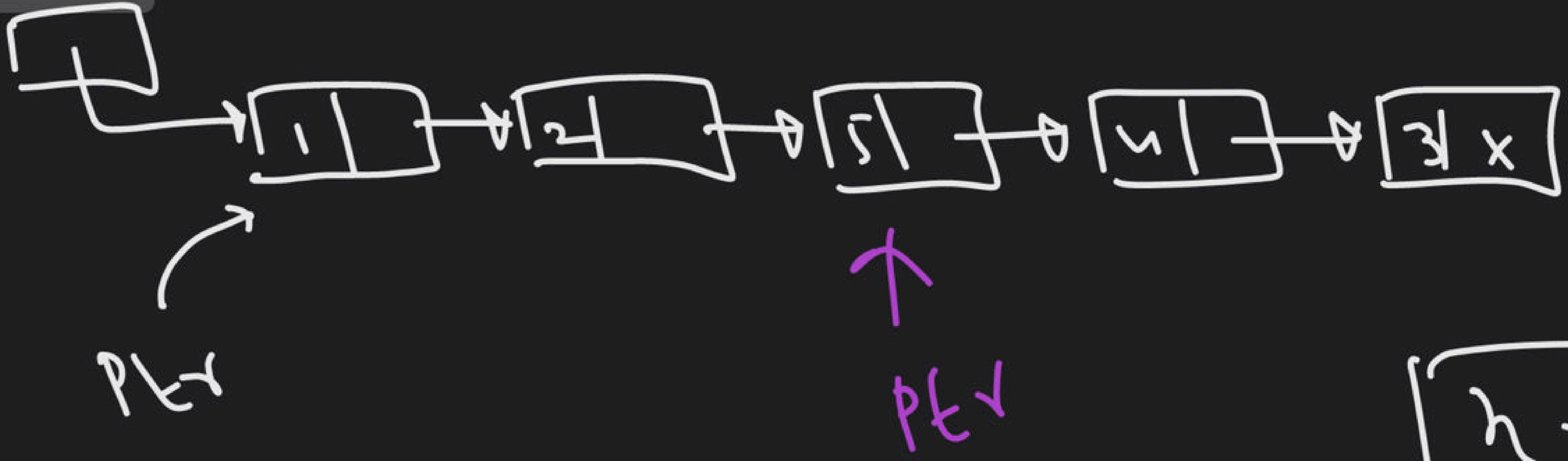




(i) Ptr = Ptr → Next } 2 times

$$\boxed{n = 5}$$

$$\frac{n}{2} = \frac{5}{2} = \boxed{2}$$

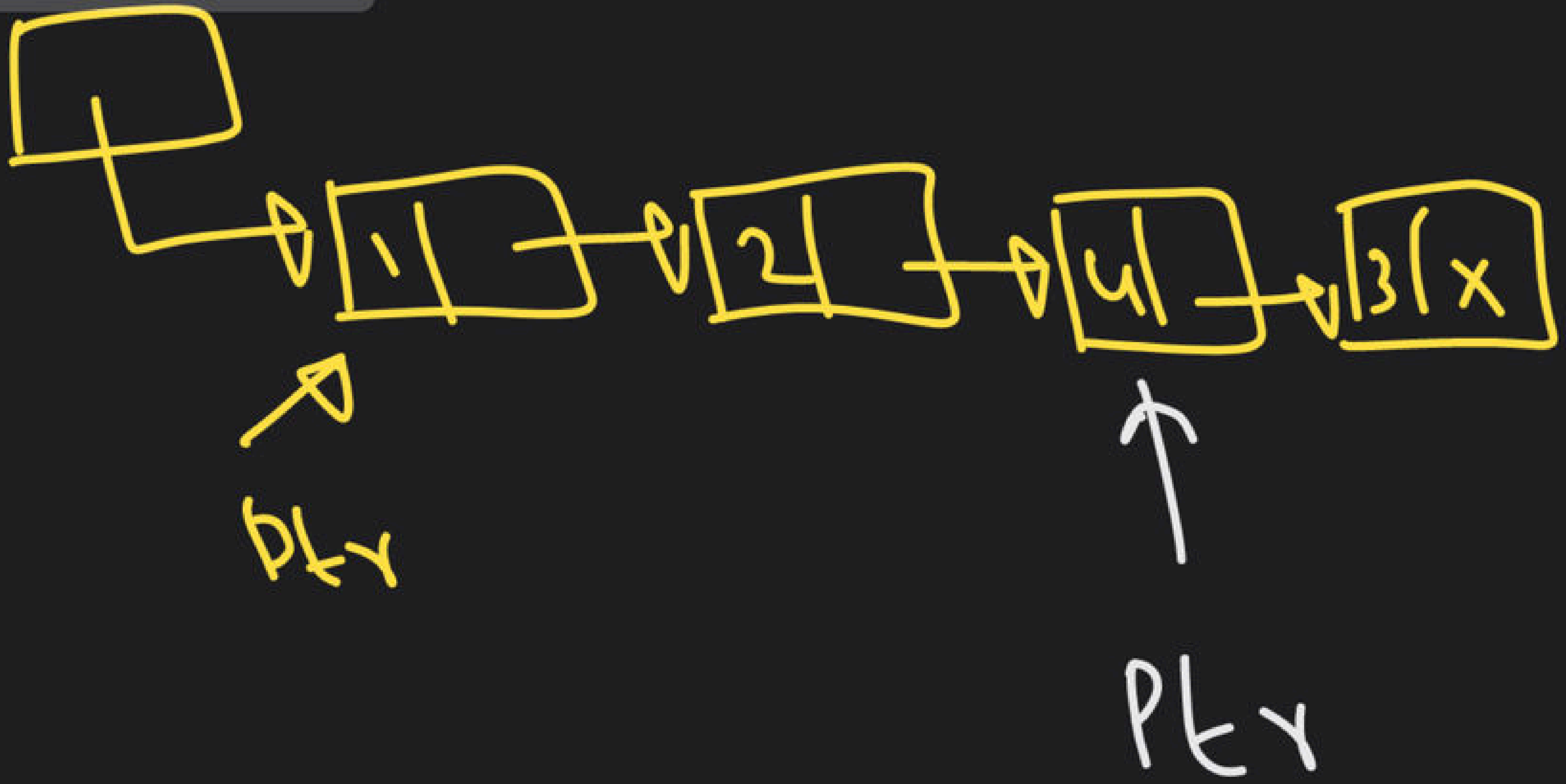


(i) $Ptr = Ptr \rightarrow Next$ } 2 times

$$\boxed{n = 5}$$

$$\frac{n}{2} = \frac{5}{2} = 2$$

3rd node



$$\frac{n}{2} = \frac{4}{2} = 2$$

Ptr = START;

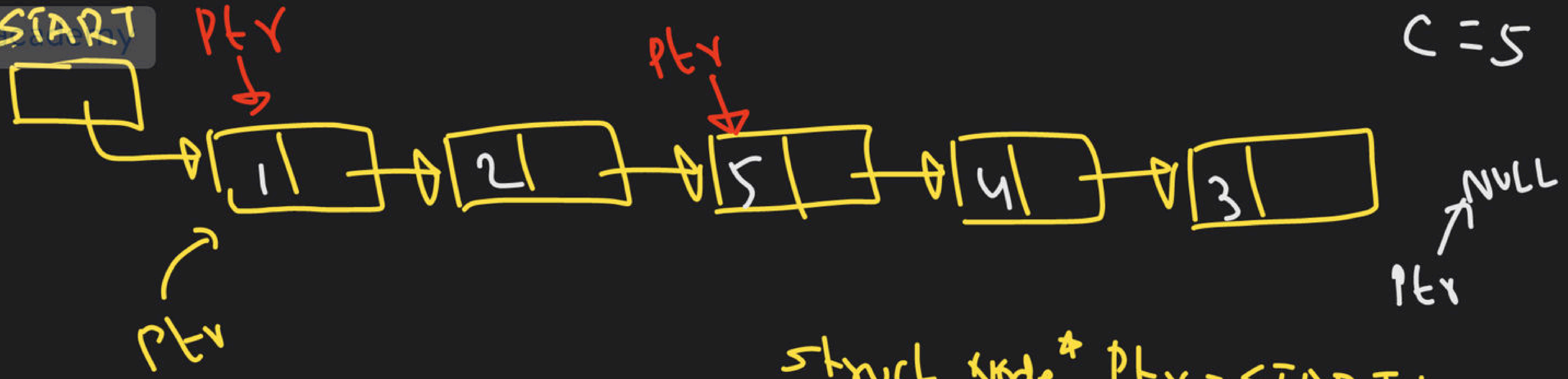
i) count no. of nodes

(ii) count = count/2;

(iii) loop => count times

Ptr = Ptr → Next

Ptr = START



c = 5

```
if (c==0)
    return;
```

```
ptr=START;
for(i=1;i<=c;i++)
        ptr=ptr→Next;
pf("%d",ptr→data);
```

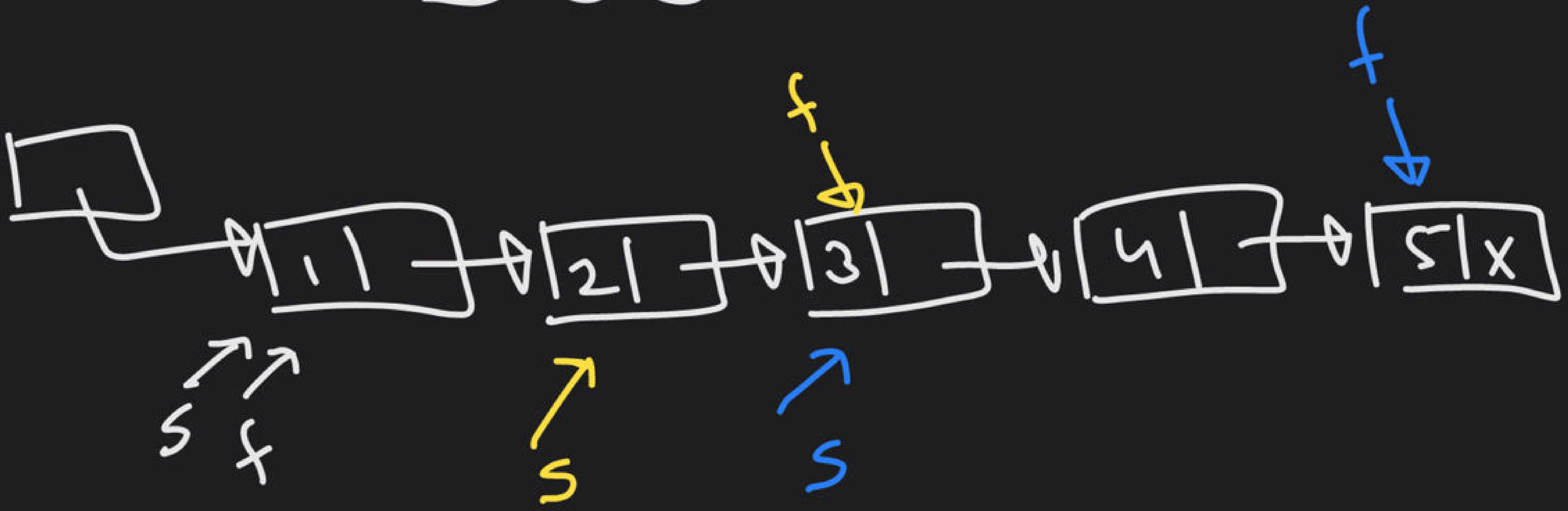
```
struct node* ptr=START;
int c=0;
while(ptr!=NULL){
            c++;
        ptr=ptr→Next;
}
```

```
c = c/2;
```

$\boxed{c = 2}$

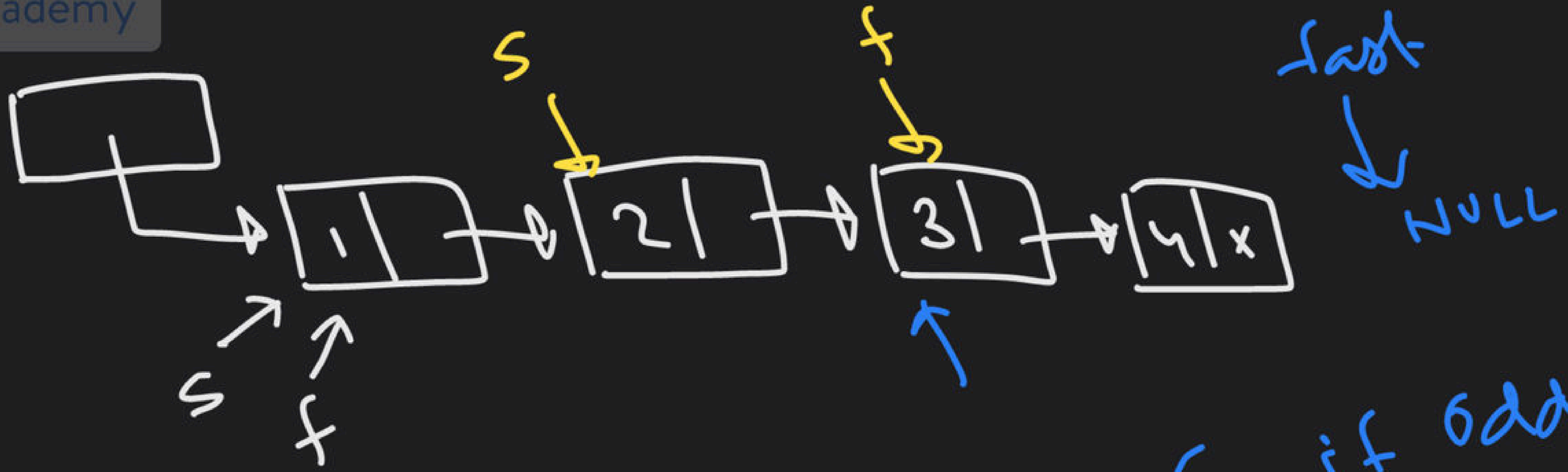
2nd Approach

fast => last node



fast->Next ==NULL

slow
↳ mid



if odd no nodes

fast ⇒ NULL

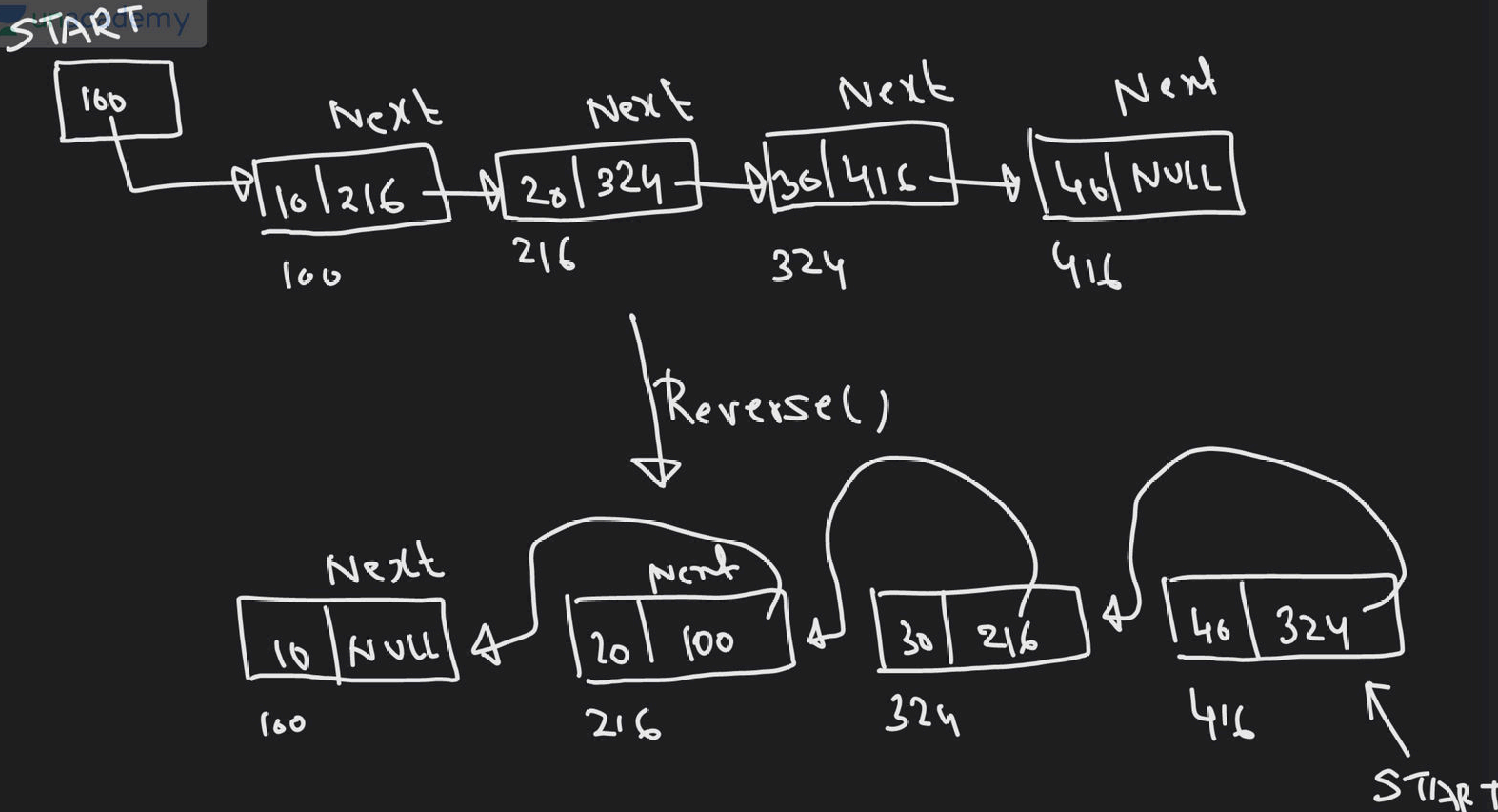
slow ⇒ mid.node

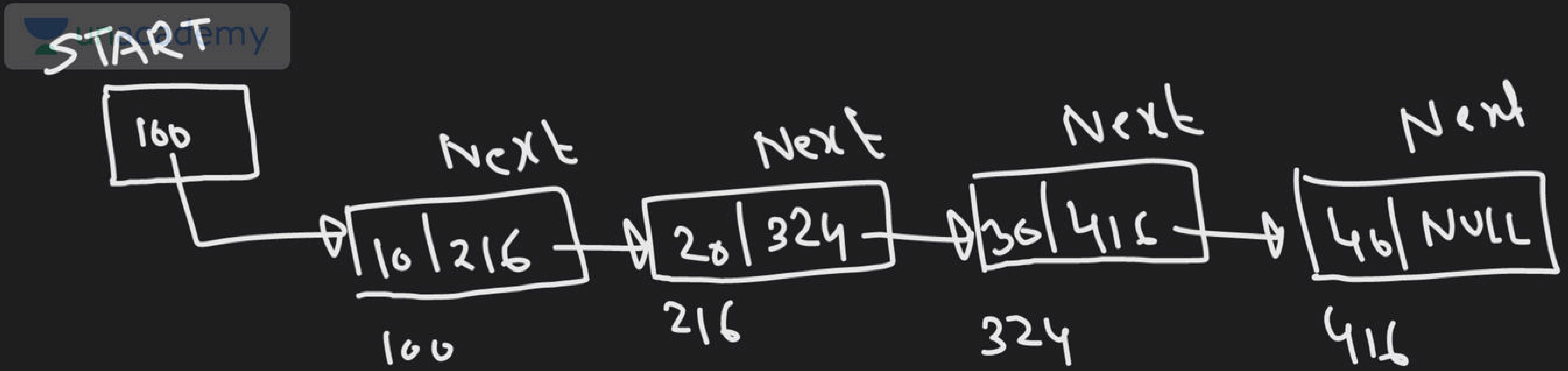
logic

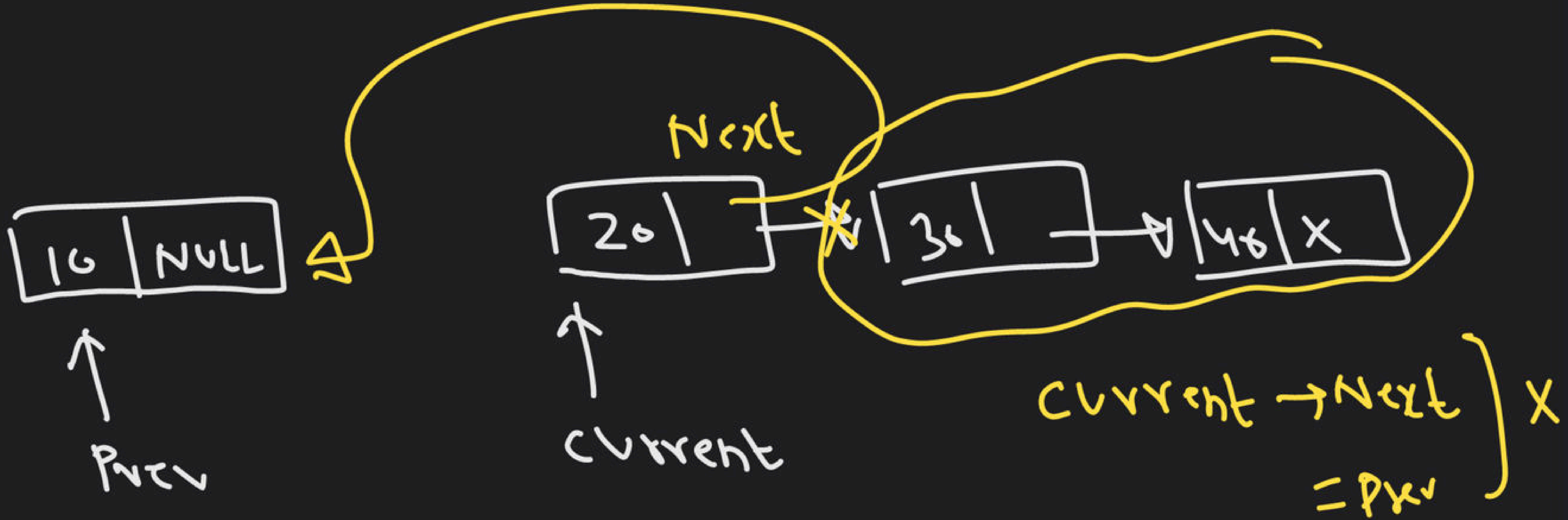
```
While (fast!=NULL && fast→Next!=NULL)
{
    slow = slow→Next;
    fast = fast→Next→Next;
}
```

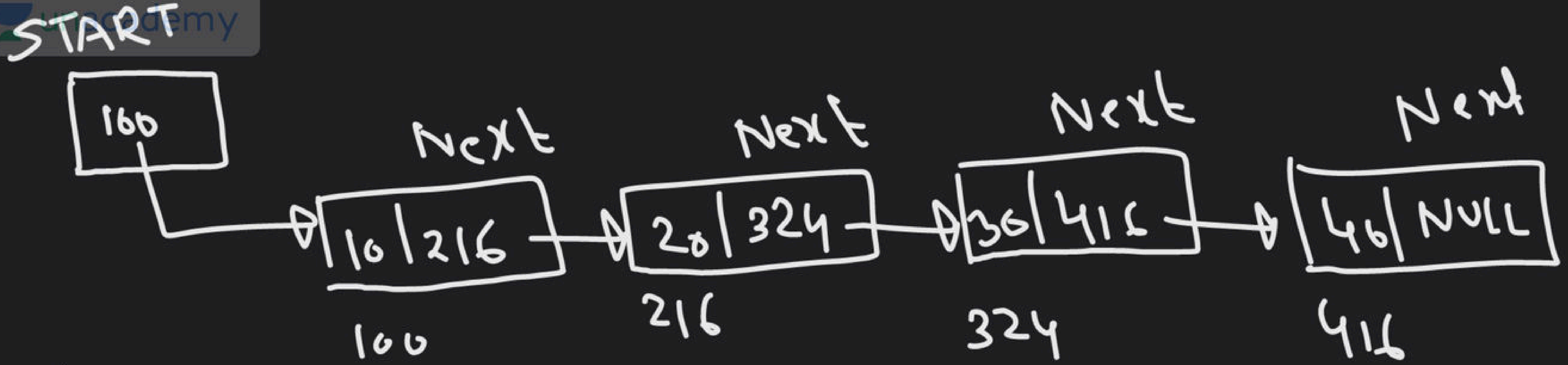
9. Reverse a Linked List

- → Printing ⇒ Recursion
- → actually reverse


START
100
Next
10
216
100
Next
20
324
216
Next
30
416
324
Next
40
NULL
416
Reverse( )
Next
10
NULL
100
Next
20
100
216
30
216
324
40
324
416
START

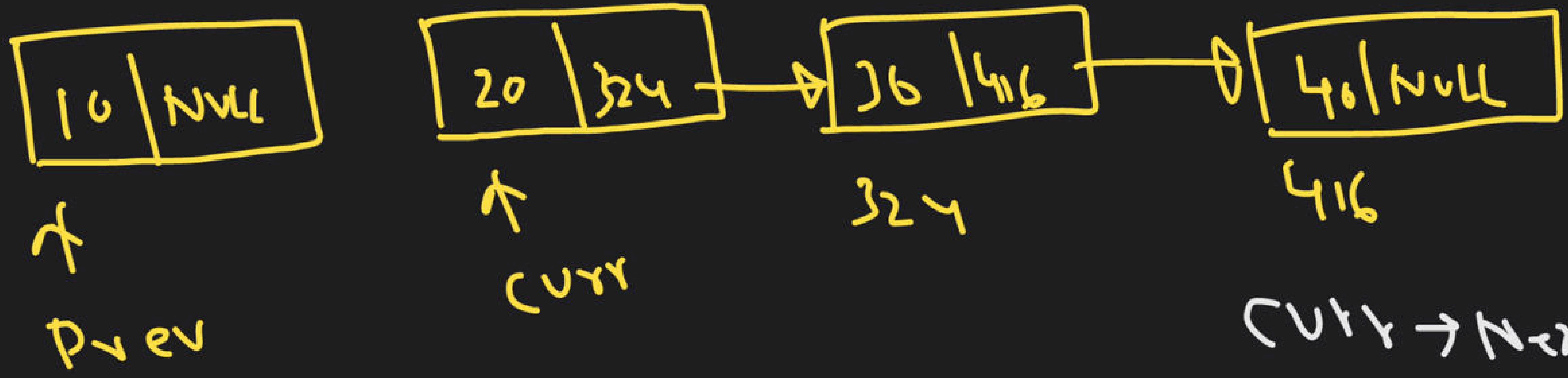

START
100
Next
Next
Next
Next
10 | 216
20 | 324
30 | 416
40 | NULL
100
216
324
416


Next
10 | NULL
20 |
30 |
40 | X
Prev
current
current → Next = Prev } X

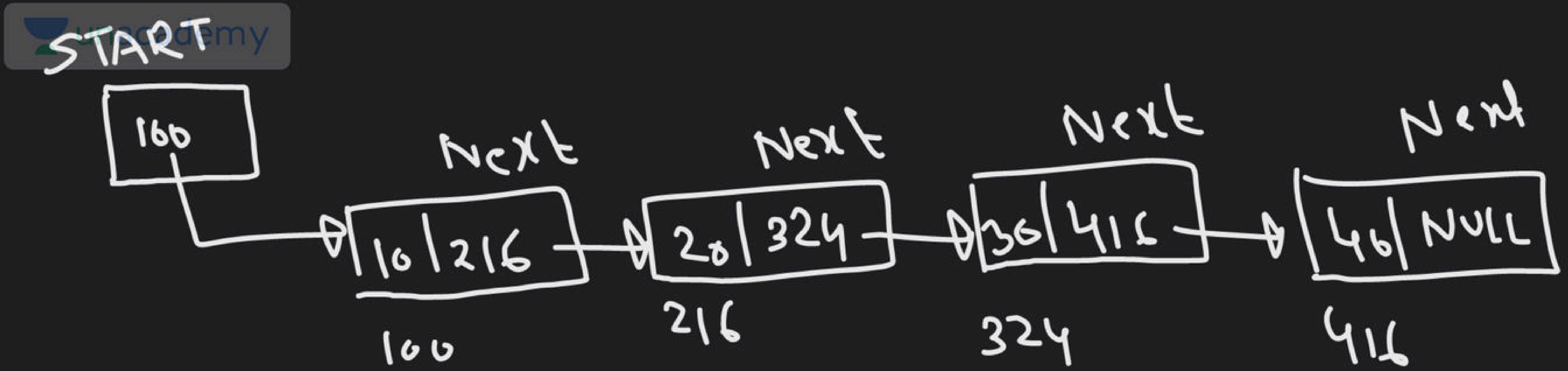


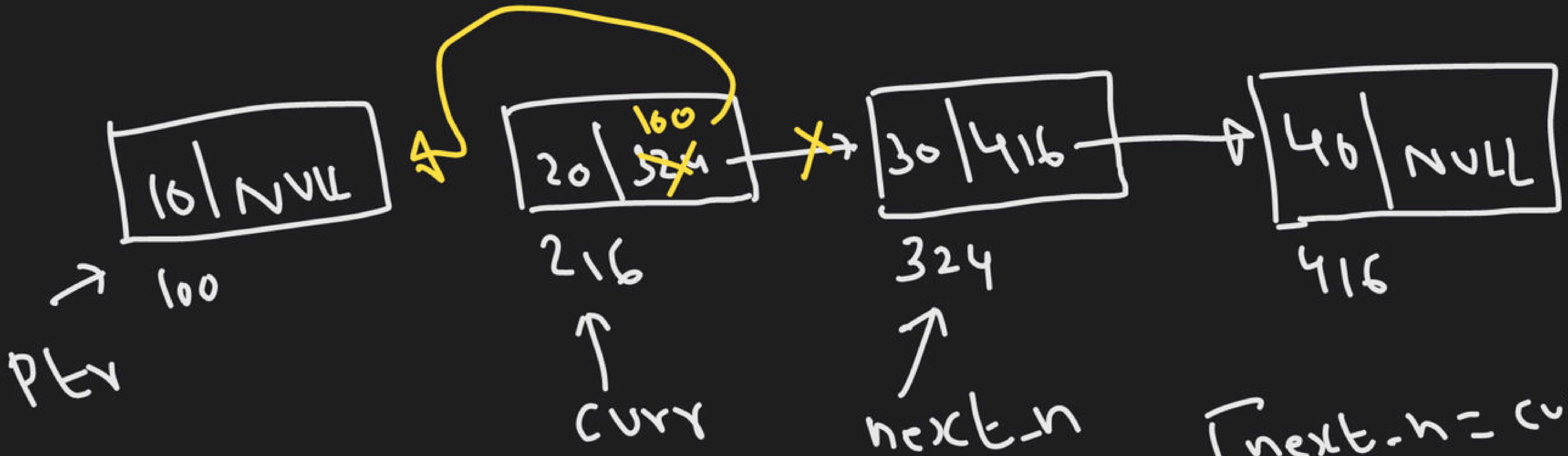
Before anything you must save the address of next node.

next_node = curr→Next

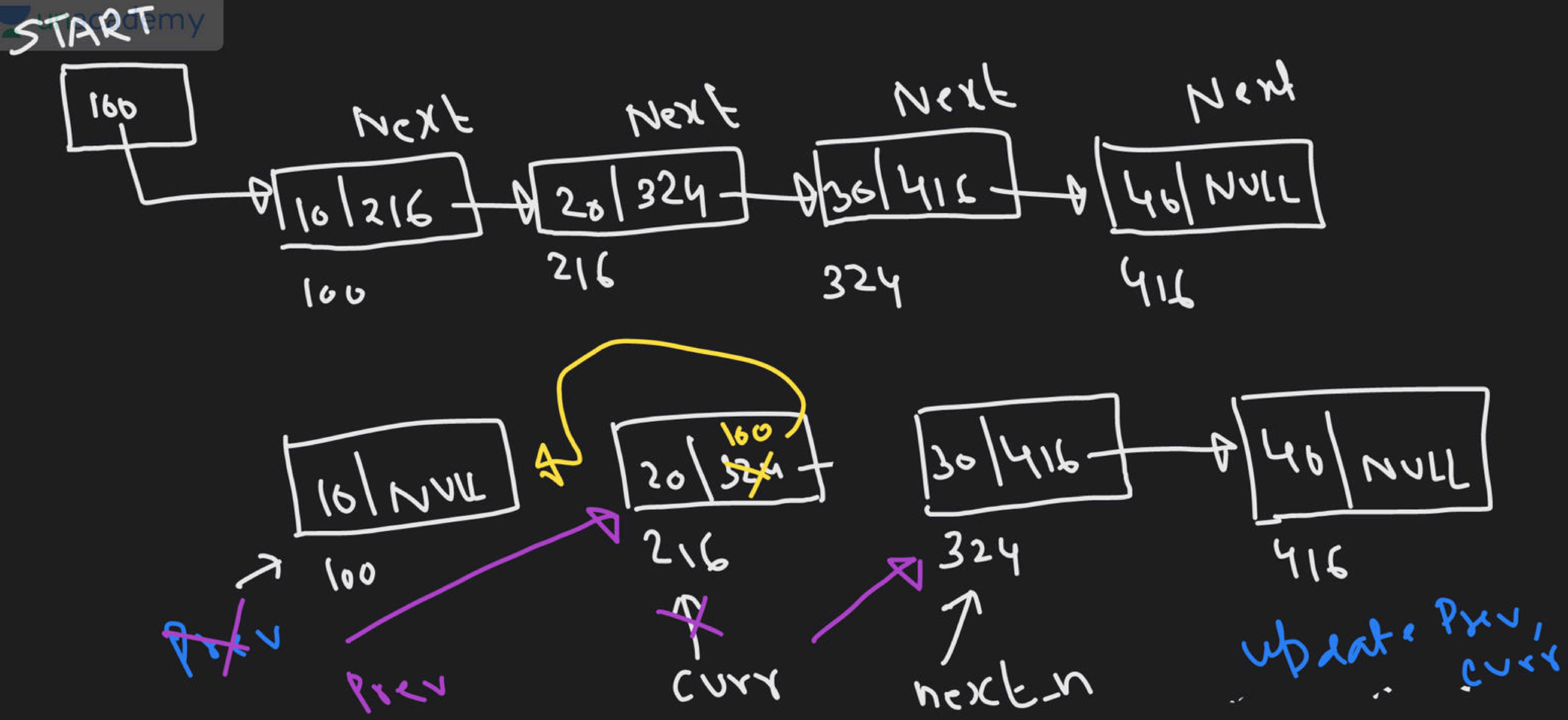


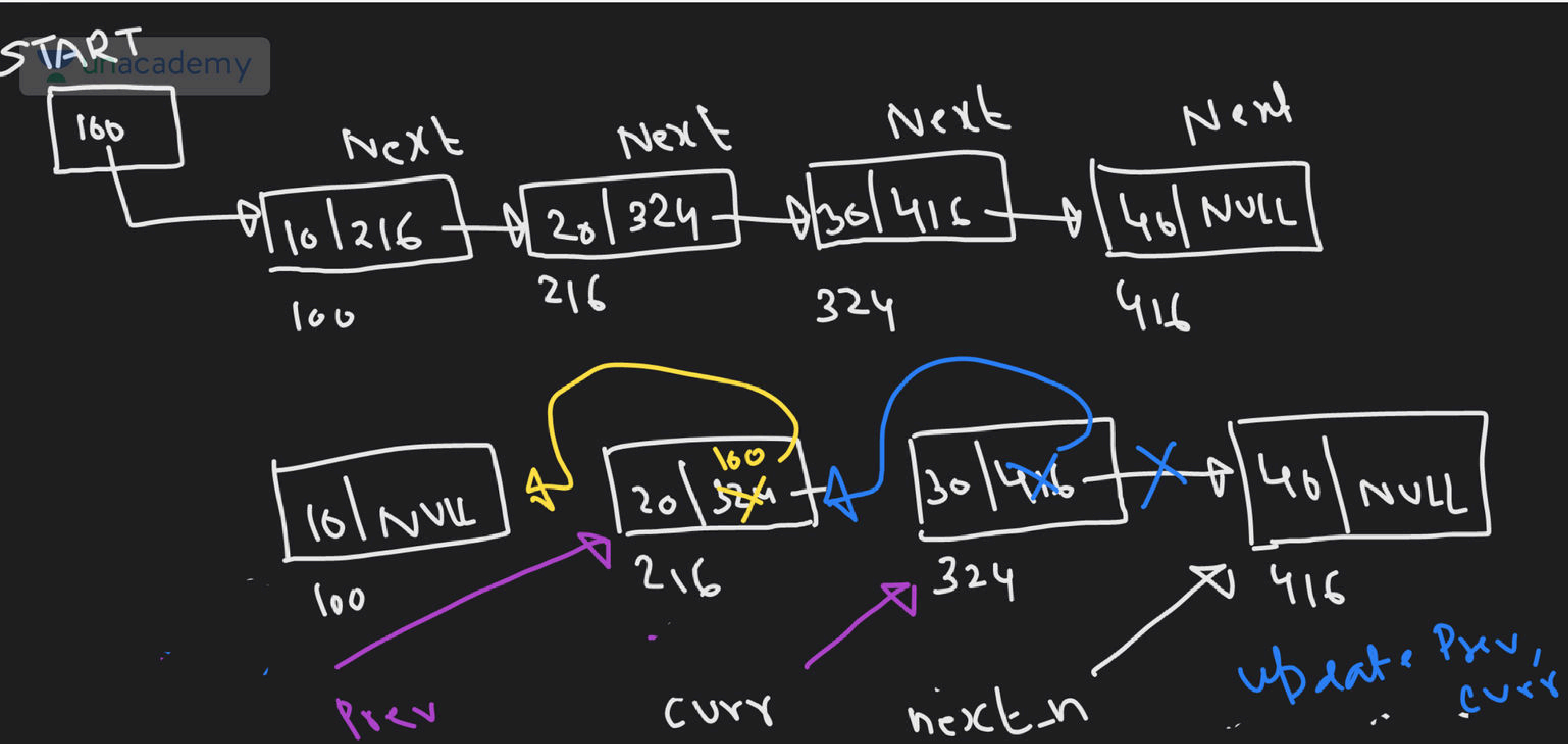
curr→Next =Prev;

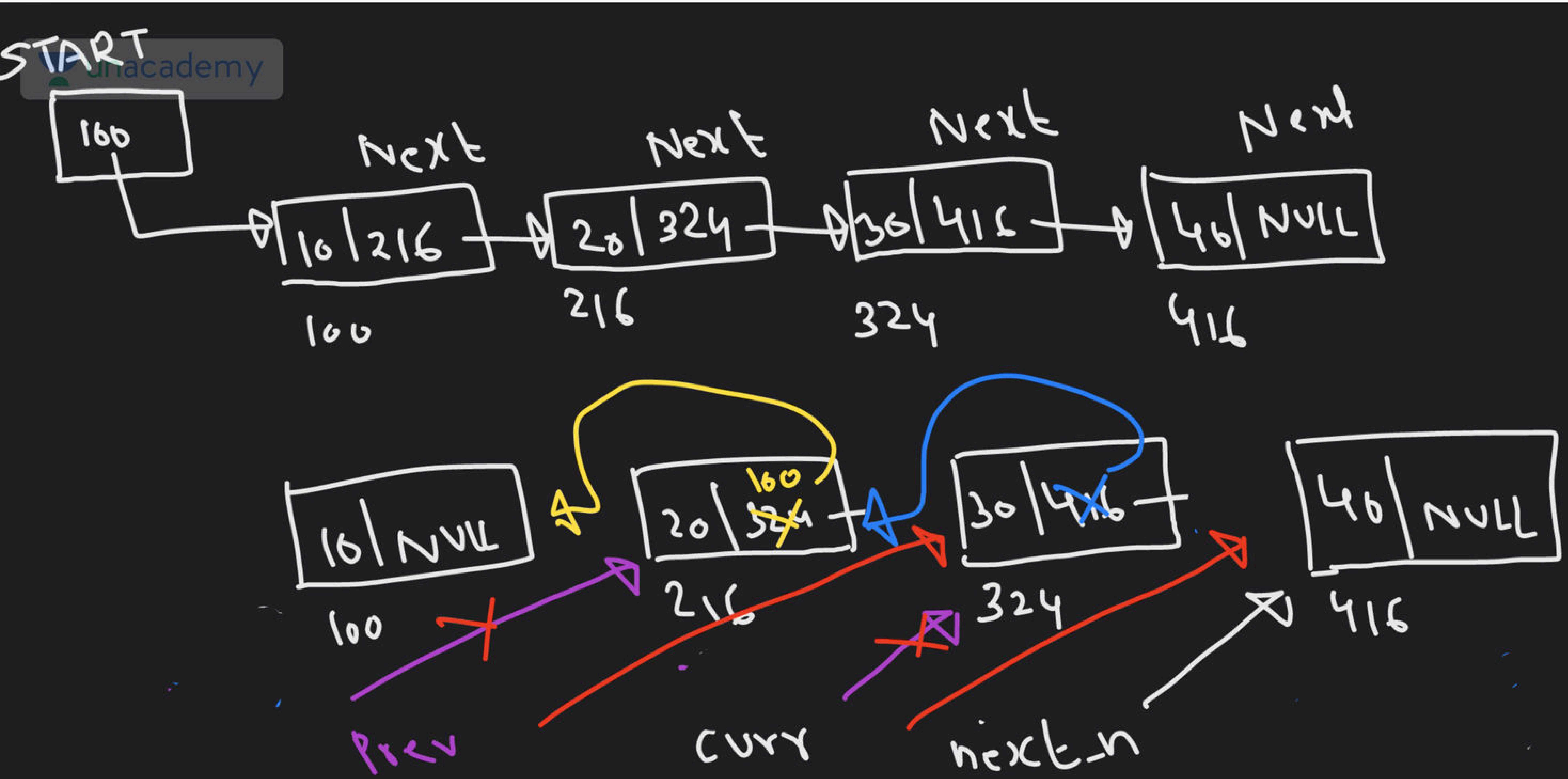


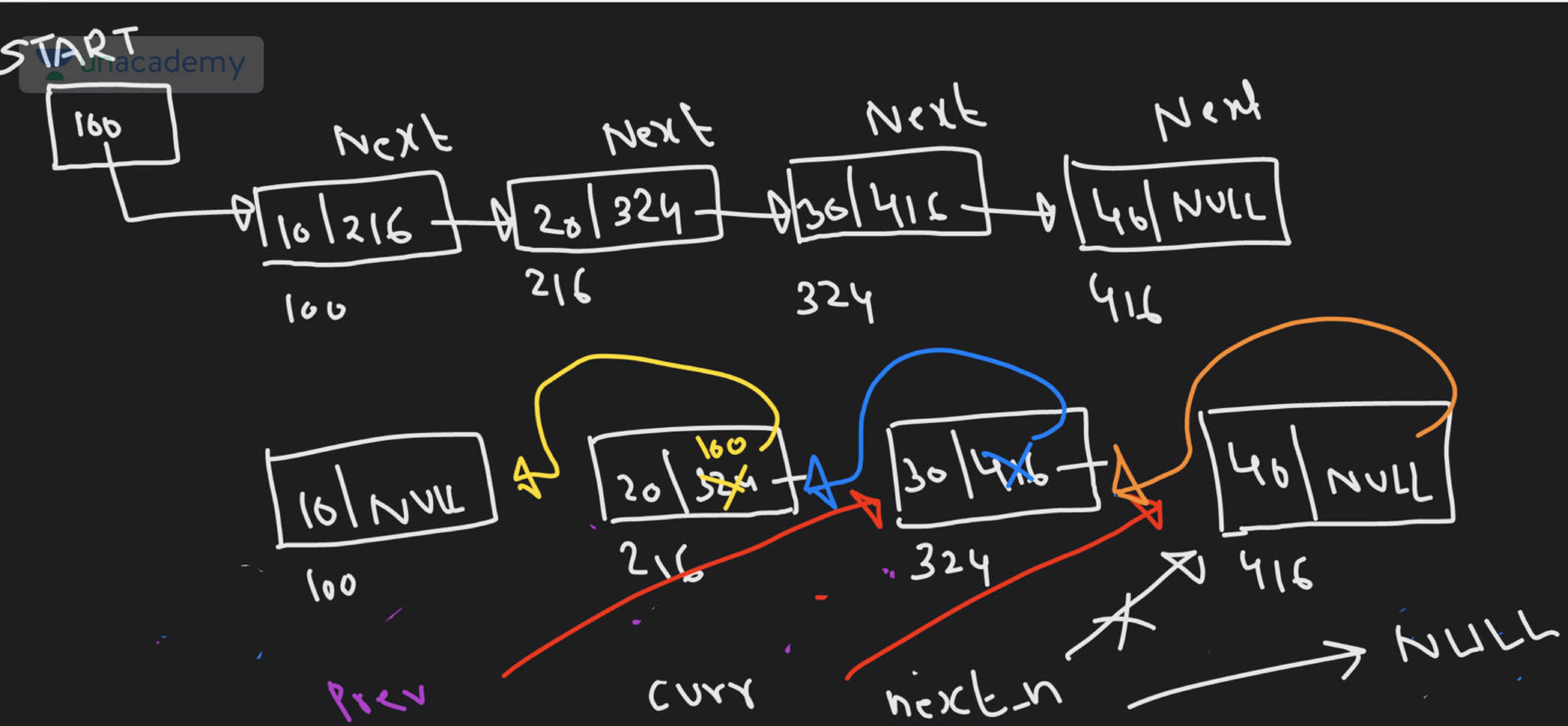


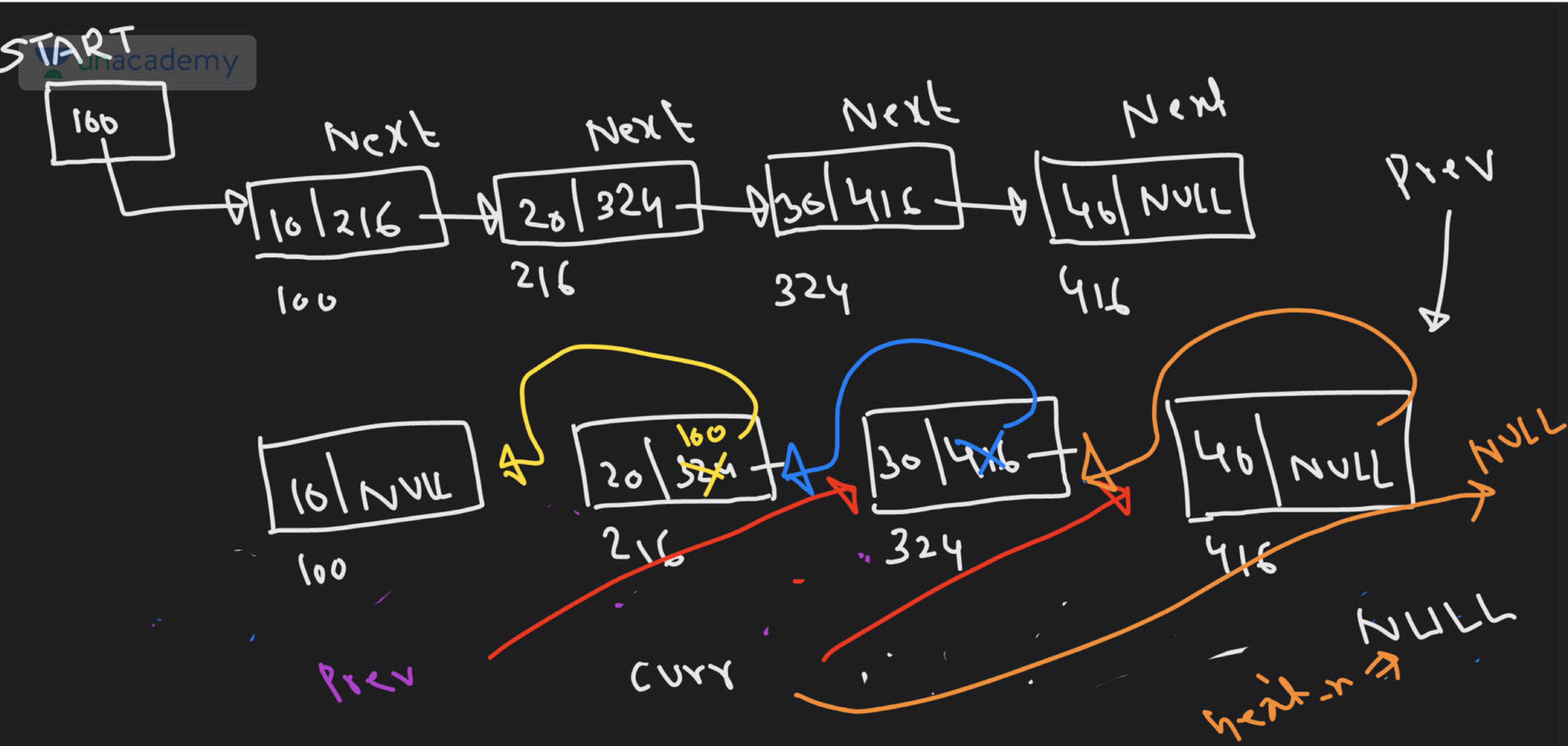
```
next_n = curr->
              next
curr->next
        = ptr;
```


START
100
Next
Next
Next
Next
10 | 216
20 | 324
30 | 416
40 | NULL
100
216
324
416
10 | NULL
20 | ~~324~~ 100
30 | 416
40 | NULL
100
216
324
416
~~Prev~~
Prev
curr
next_n
update Prev,
curr


START
100
Next
Next
Next
Next
10 | 216
20 | 324
30 | 416
40 | NULL
100
216
324
416
10 | NULL
20 | 324
100
30 | 416
40 | NULL
100
216
324
416
Prev
curr
next_n
update Prev,
curr

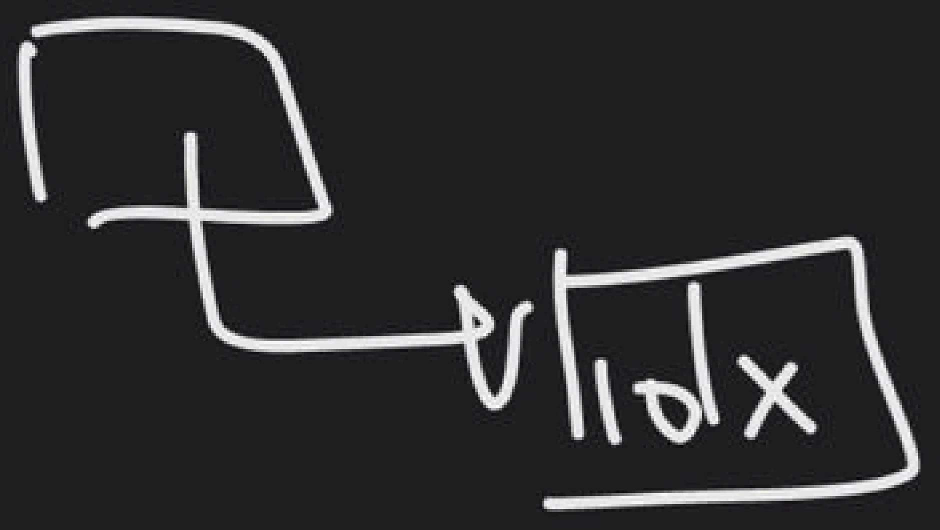

START
100
Next
Next
Next
Next
10 | 216
20 | 324
30 | 416
40 | NULL
100
216
324
416
10 | NULL
20 | ~~324~~ 100
30 | ~~416~~
40 | NULL
100
216
324
416
Prev
curr
next_n


START
100
Next
Next
Next
Next
10 | 216
20 | 324
30 | 416
40 | NULL
100
216
324
416
10 | NULL
20 | 324 100
30 | 416
40 | NULL
100
216
324
416
prev
curr
next_n
NULL


START
100
Next
Next
Next
Next
10 | 216
20 | 324
30 | 416
40 | NULL
100
216
324
416
Prev
10 | NULL
20 | ~~324~~ 100
30 | ~~416~~
40 | NULL
NULL
100
216
324
416
Prev
Curr
NULL
Next_n →

1 if LL is empty → START = = NULL

2 1 Node

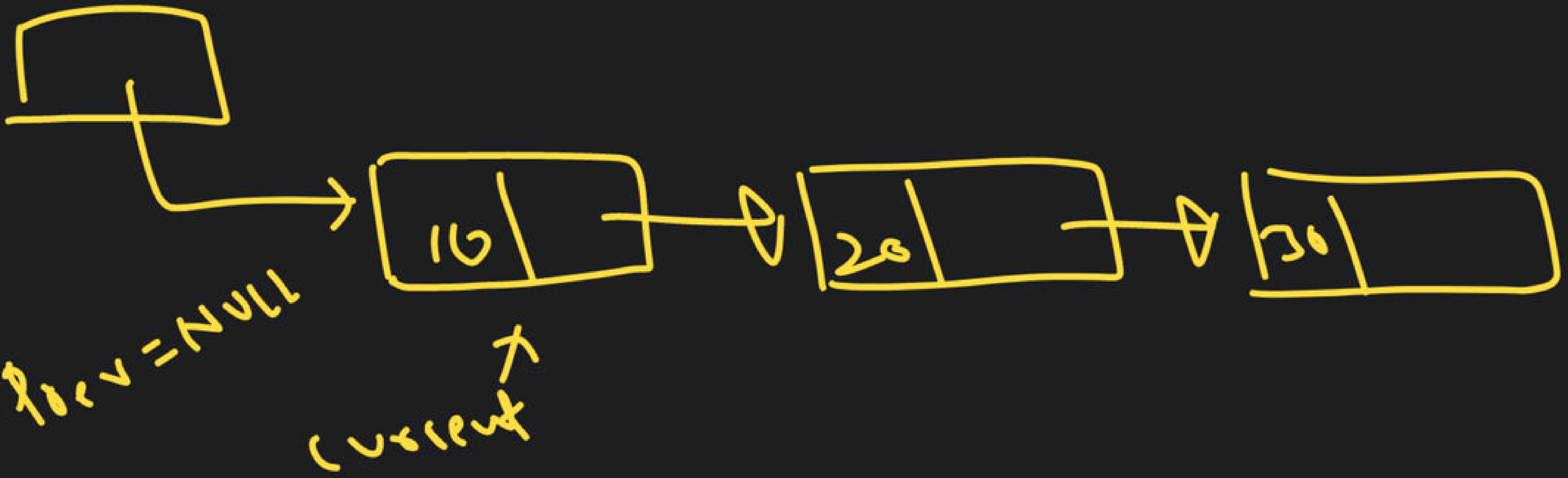


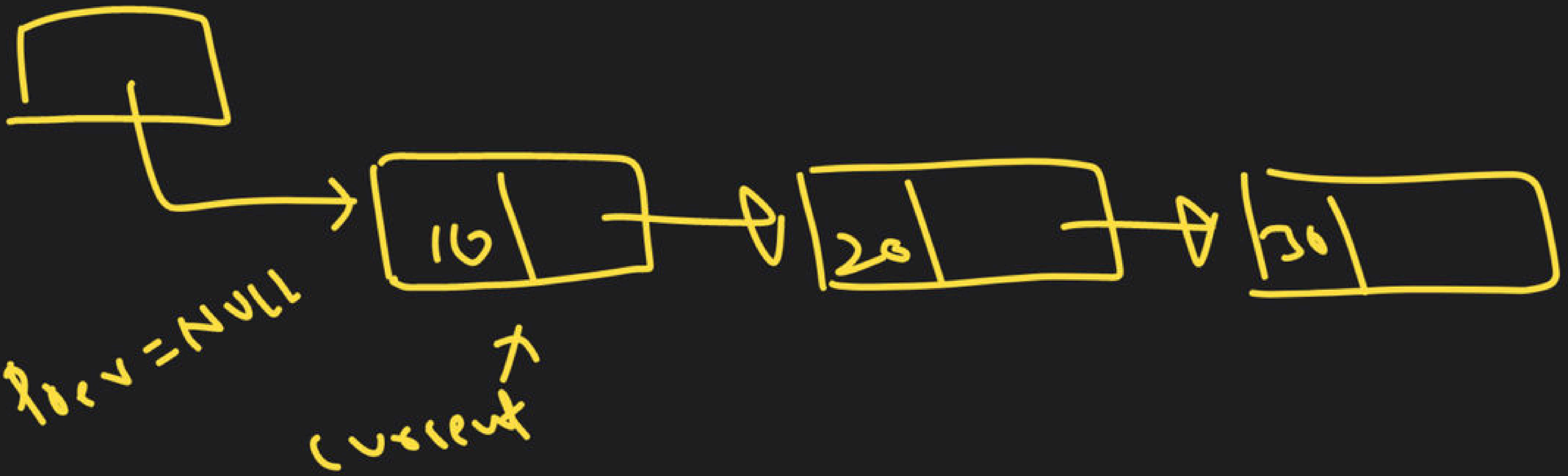
→ START → Next
= = NULL

do Nothing

```
Struct Node *Prev, *Nex, *current;
if ( START == NULL || START->Next == NULL
                  return;

    current = START;
    Prev = NULL;
```

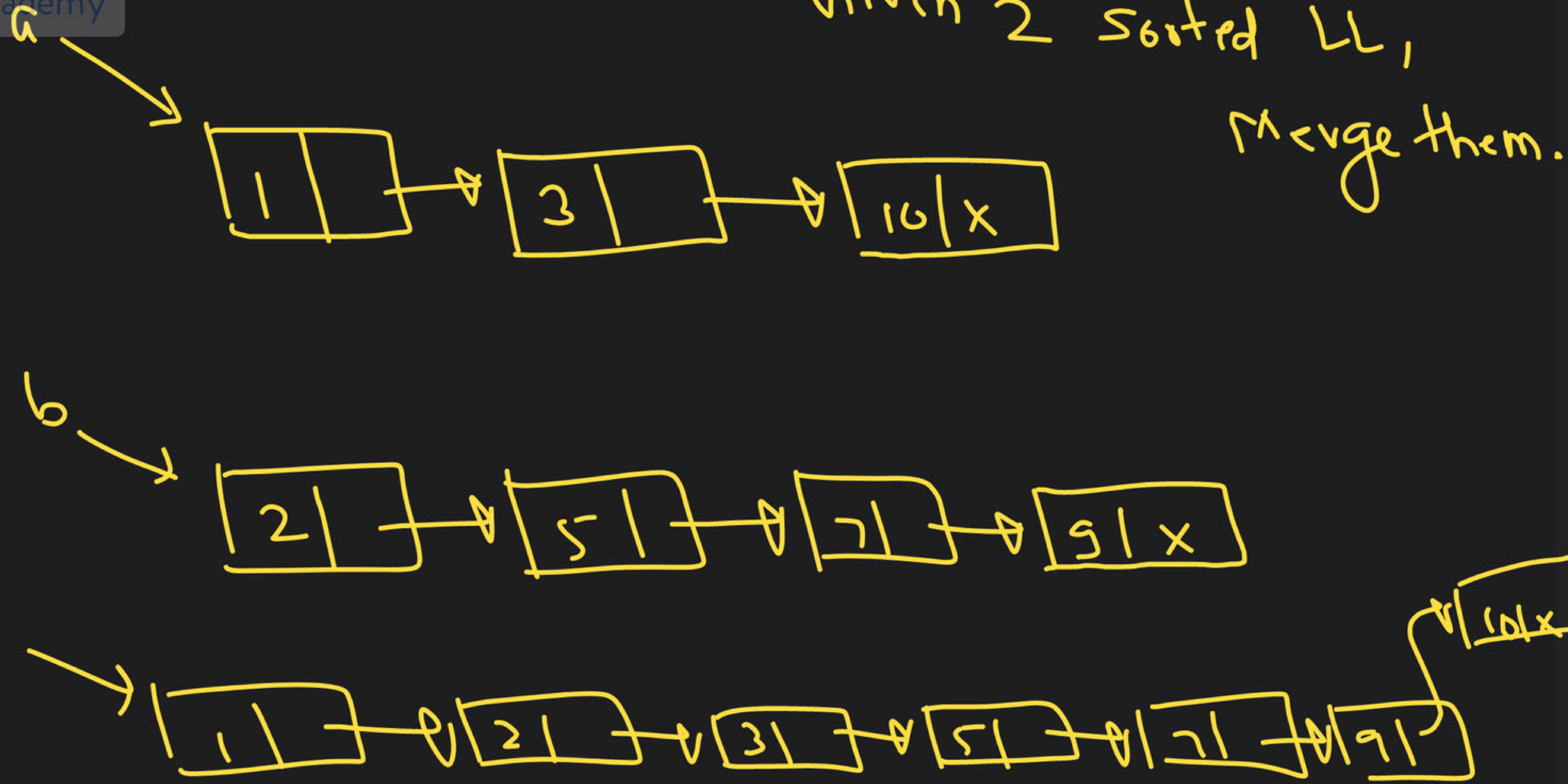
```
struct Node *Prev,*Nex ,*current;
if ( START = = NULL || START →Next = =NULL
        return;

current = START;
Prev = NULL;
```





```
while(current! = NULL)
 {
     Nex = curr →Next
     current→Next = Prev;
     Prev = current;
      current = Nex;
 }
return Prev;
```

Given 2 sorted LL,
Merge them.



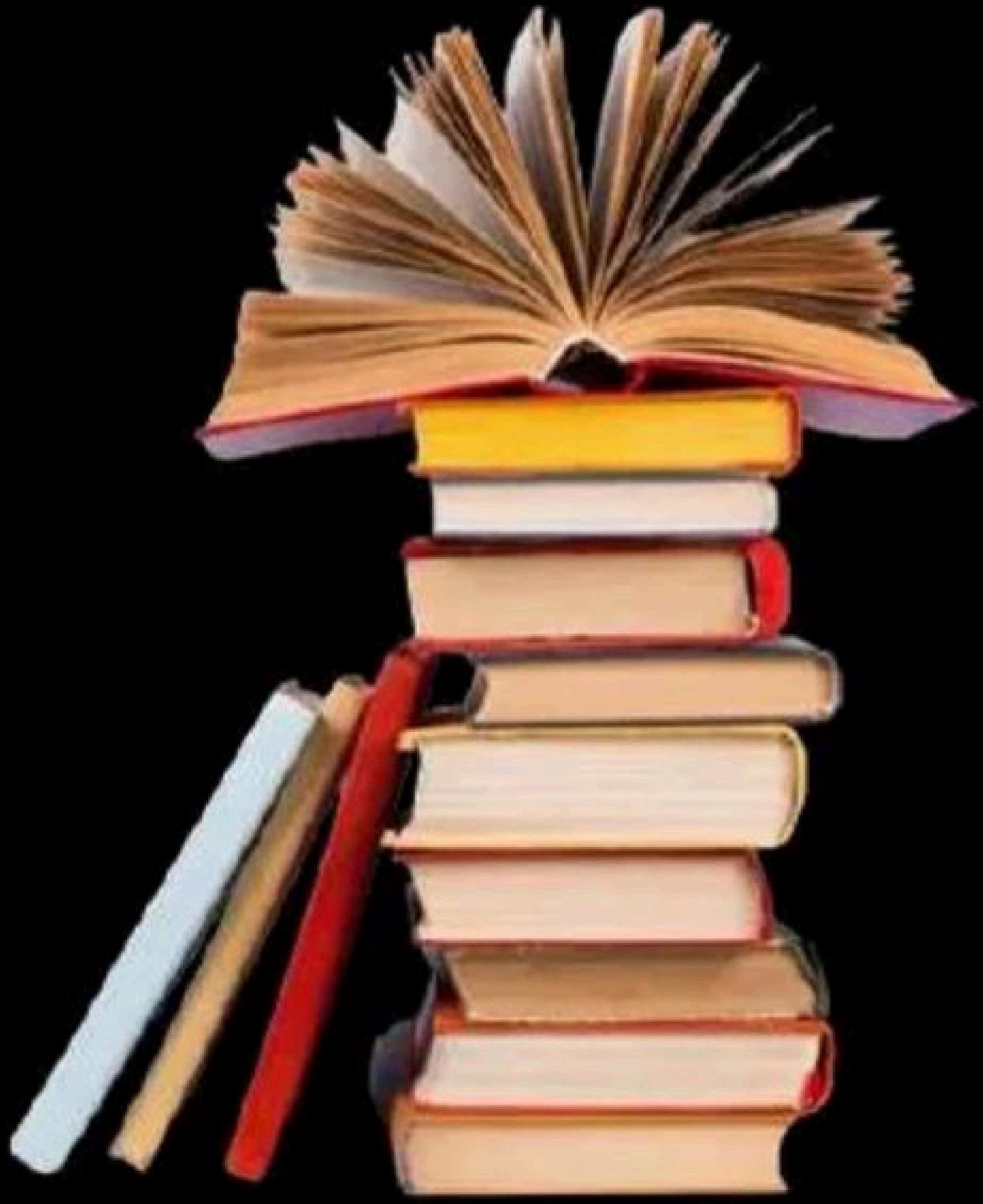
Q Given a LL, detect a loop.

Types of LL

Topics

to be covered

1. Linked List



unacademy

THANK YOU!

Here’s to a cracking journey ahead!