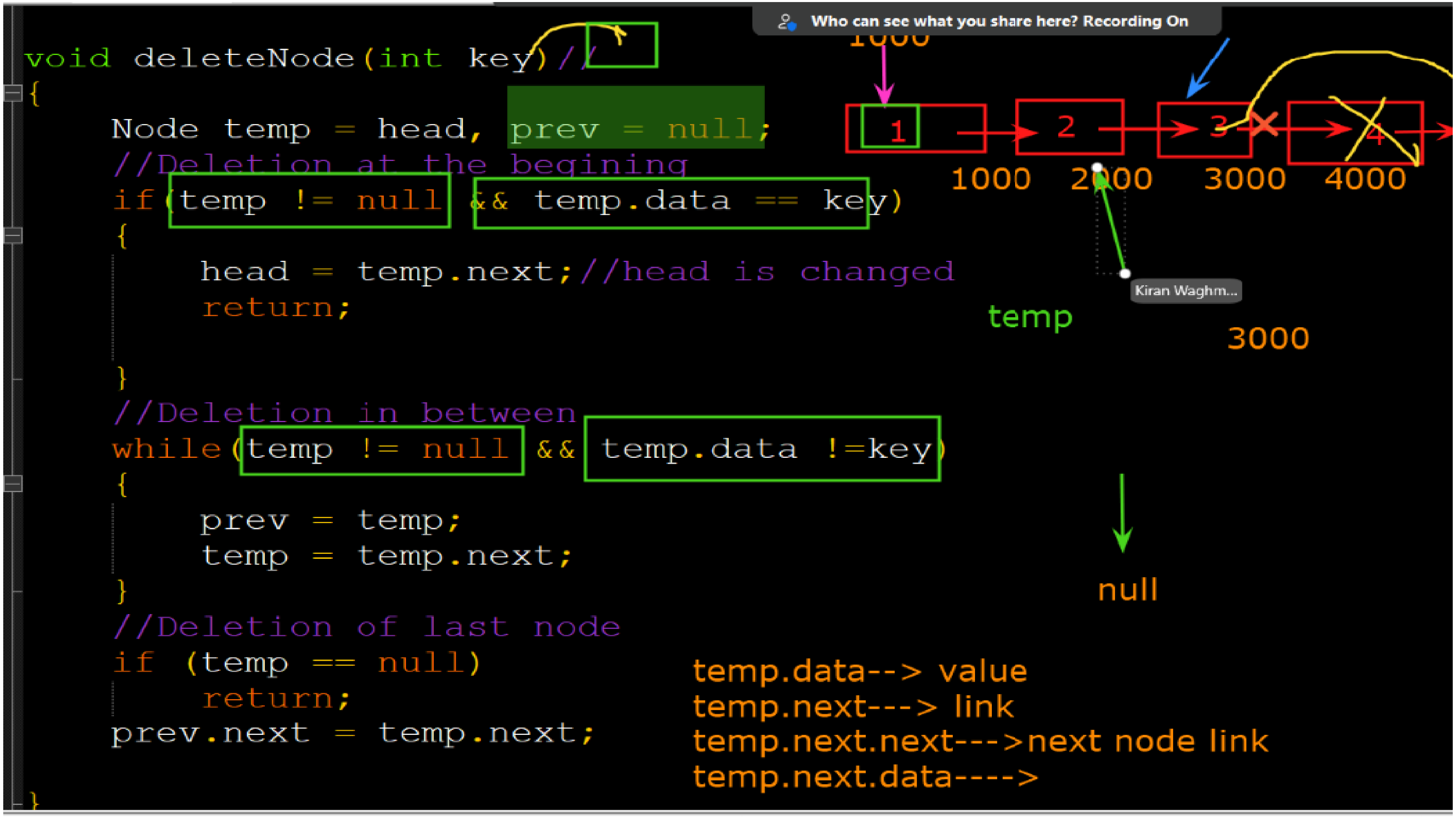
Sep22 : Day 7

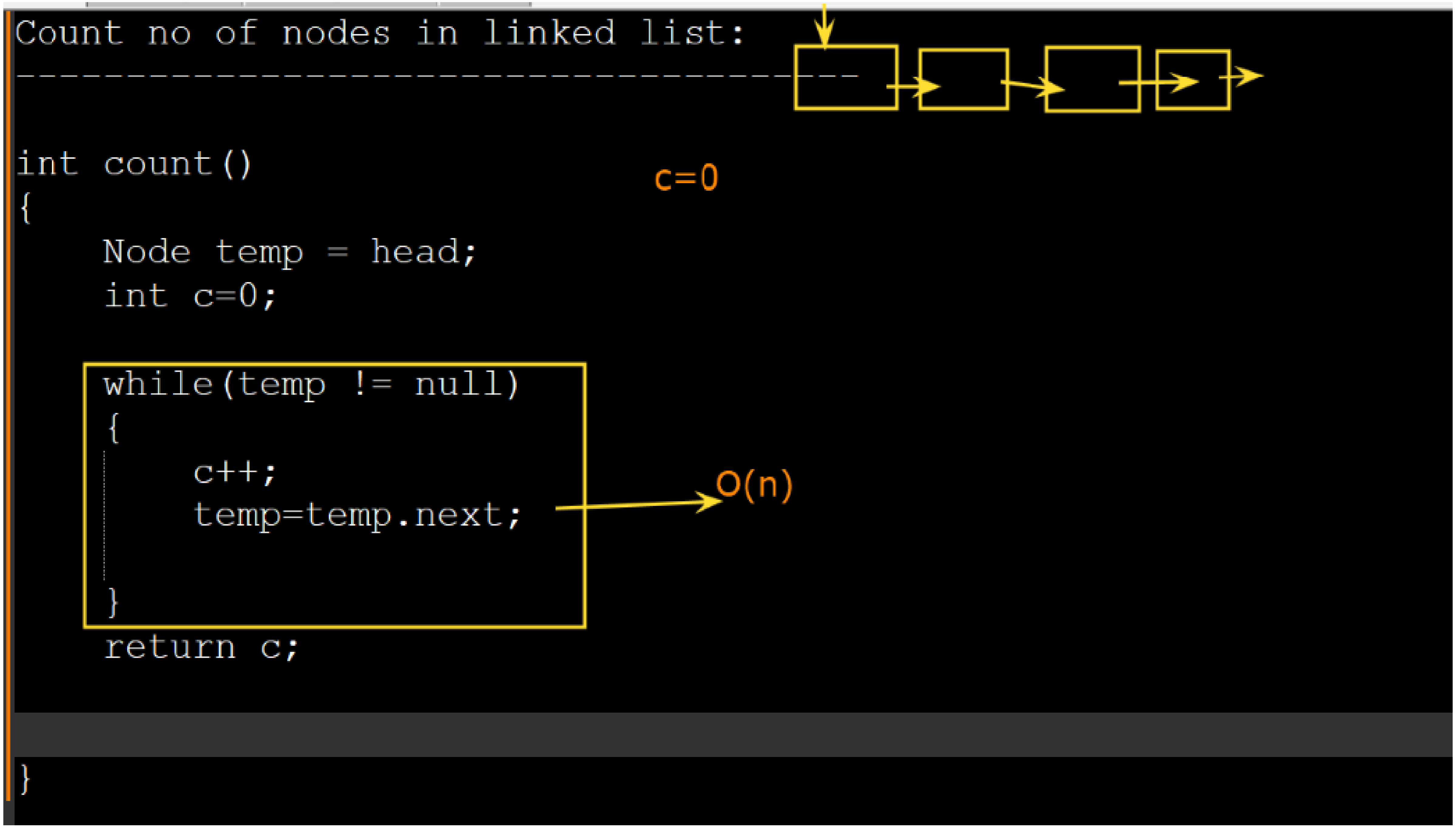
Kiran Waghmare

CDAC Mumbai

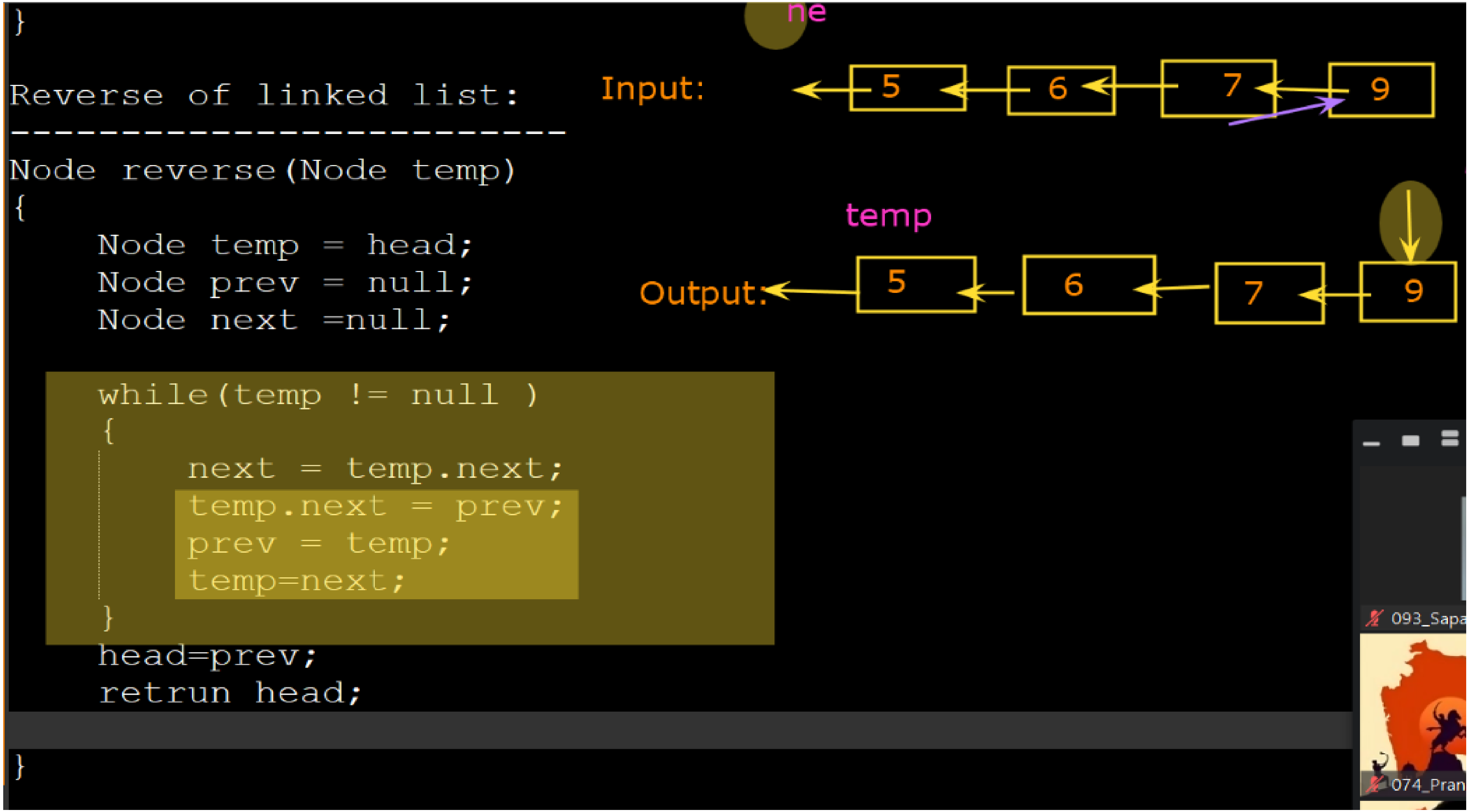


```
void deleteNode(int key)//
{
    Node temp = head, prev = null;
    //Deletion at the begining
    if(temp != null && temp.data == key)
    {
        head = temp.next;//head is changed
        return;

    }
    //Deletion in between
    while(temp != null && temp.data !=key)
    {
        prev = temp;
        temp = temp.next;
    }
    //Deletion of last node
    if (temp == null)
        return;
    prev.next = temp.next;

}
```

temp.data--> value
temp.next---> link
temp.next.next--->next node link
temp.next.data---->

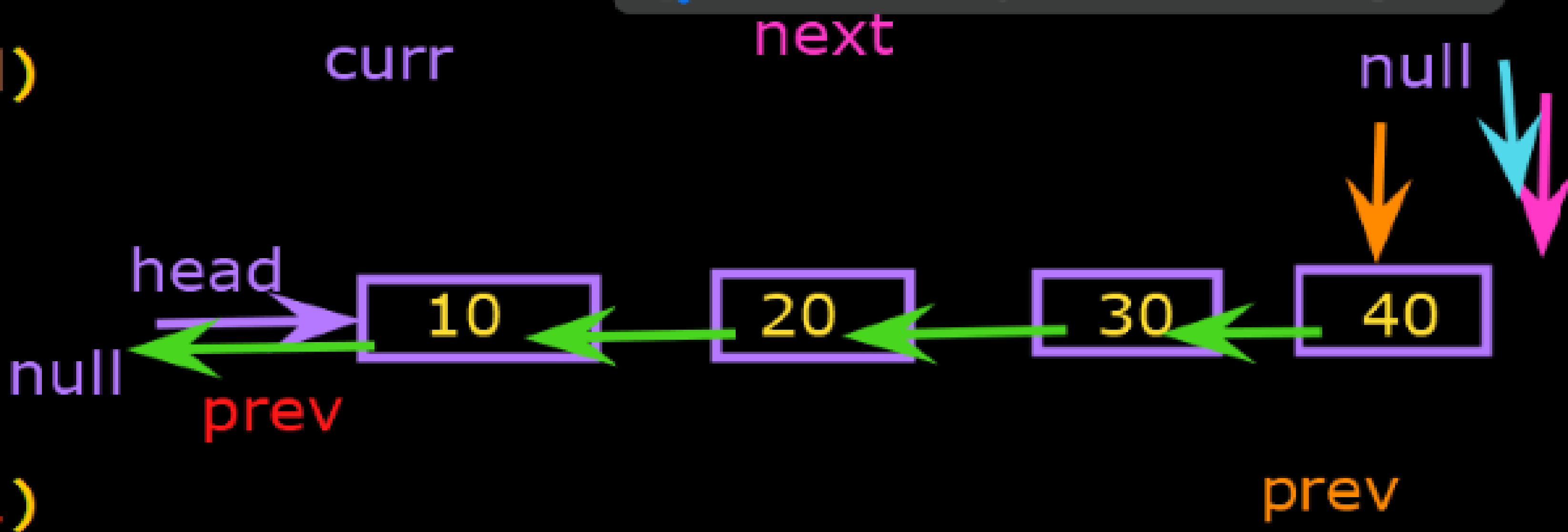


```
Count no of nodes in linked list:
-----------------------------------------

int count()
{
    Node temp = head;
    int c=0;

    while(temp != null)
    {
        c++;
        temp=temp.next;

    }
    return c;


}
```



```
}

Reverse of linked list:
---------------------------
Node reverse(Node temp)
{
    Node temp = head;
    Node prev = null;
    Node next =null;

    while(temp != null )
    {
        next = temp.next;
        temp.next = prev;
        prev = temp;
        temp=next;
    }
    head=prev;
    retrun head;

}
```

```
Node reverse(Node head)
{
    Node prev = null;
    Node curr = head;
    Node next = null;

    while(curr != null)
    {
        next = curr.next;
        curr.next = prev;
        prev = curr;
        curr = next;

    }
    head = prev;
    return head;
}
```

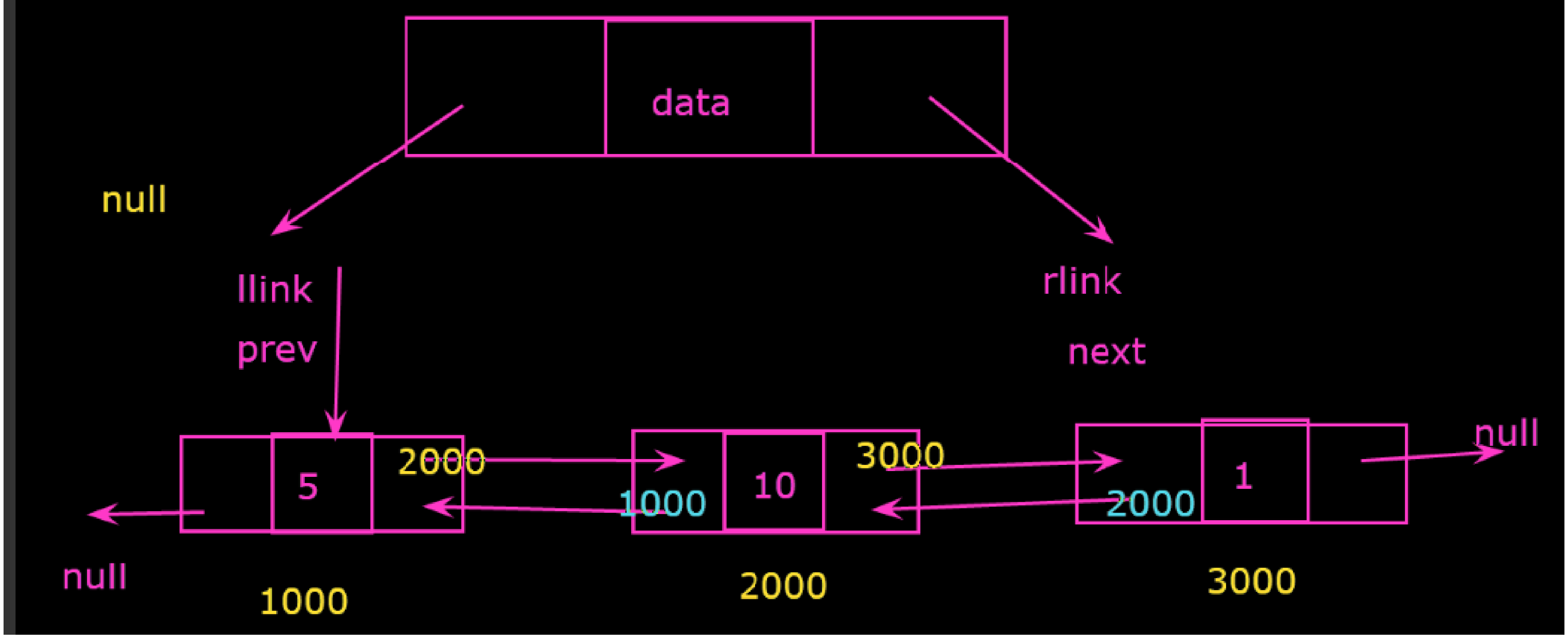


next = curr.next

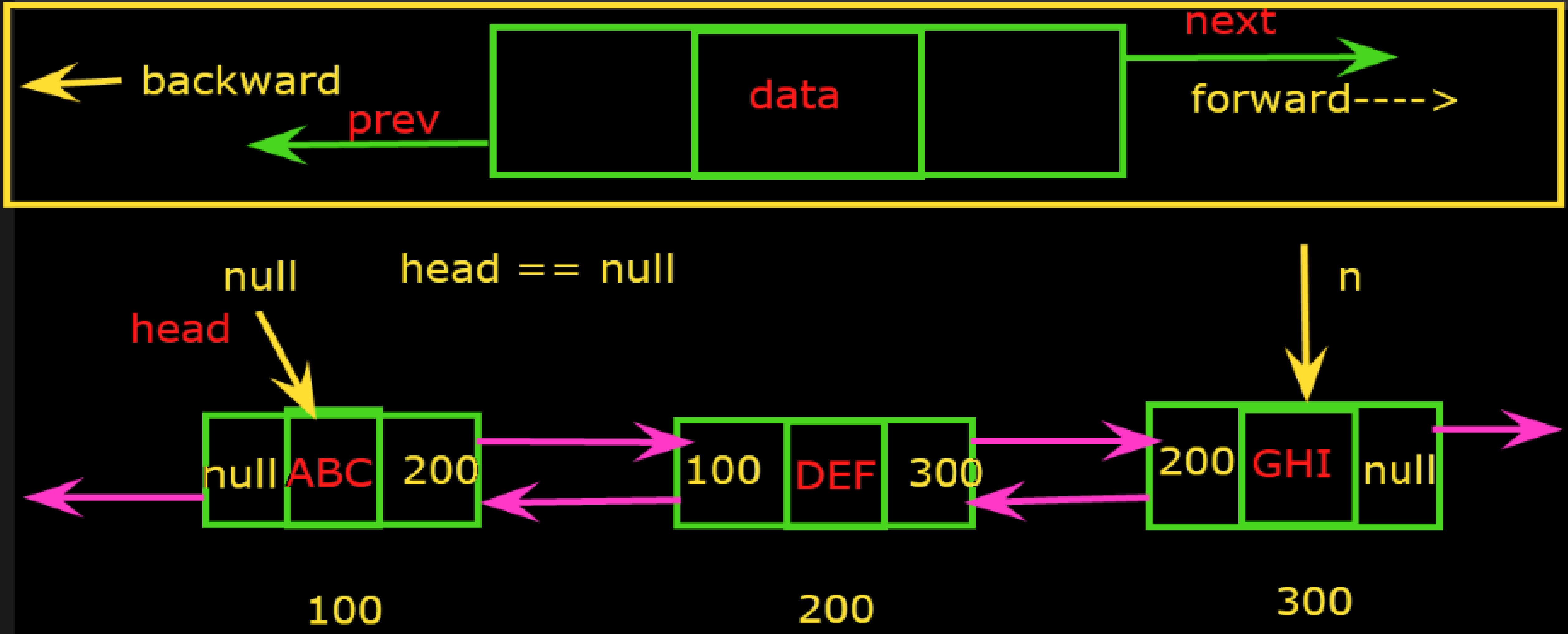
curr.next = prev

curr = next

```
Doubly Linked List:
--------------------
```

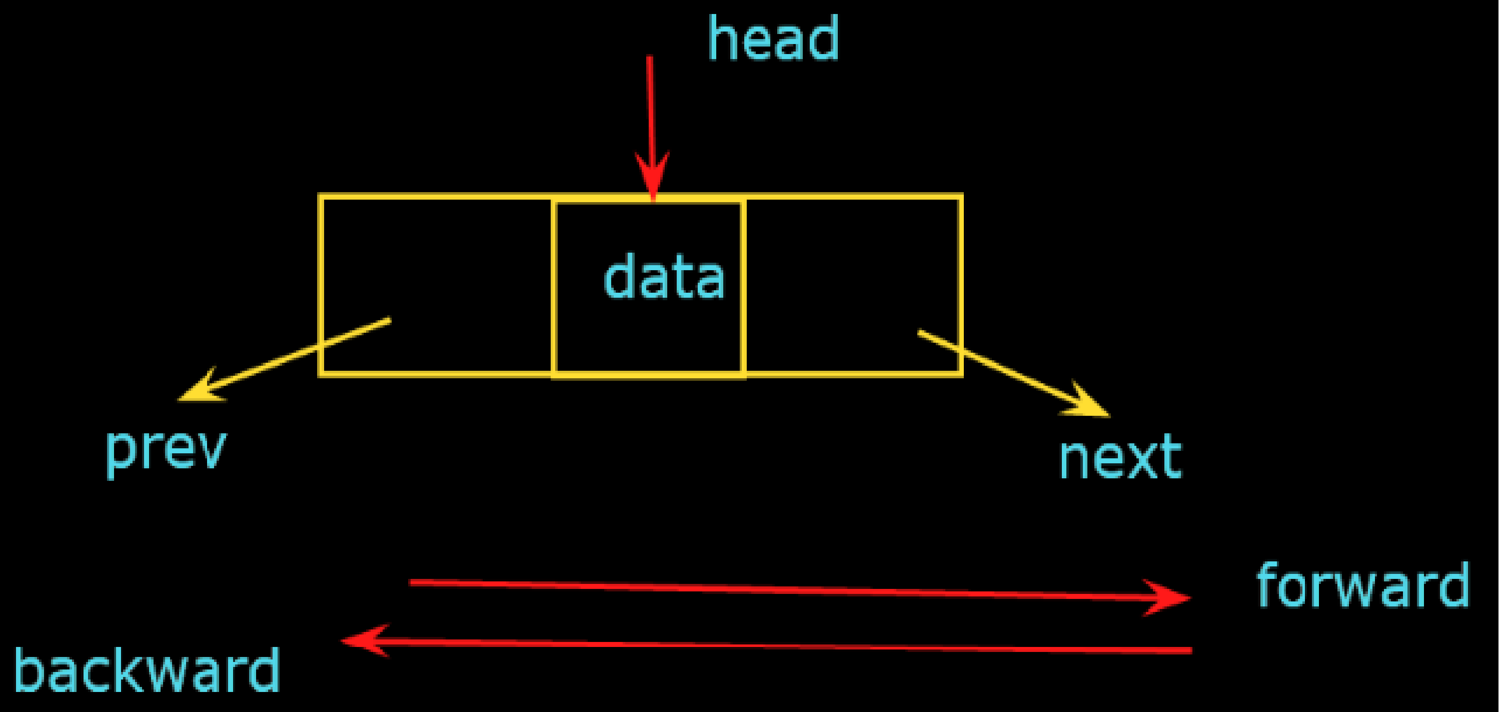


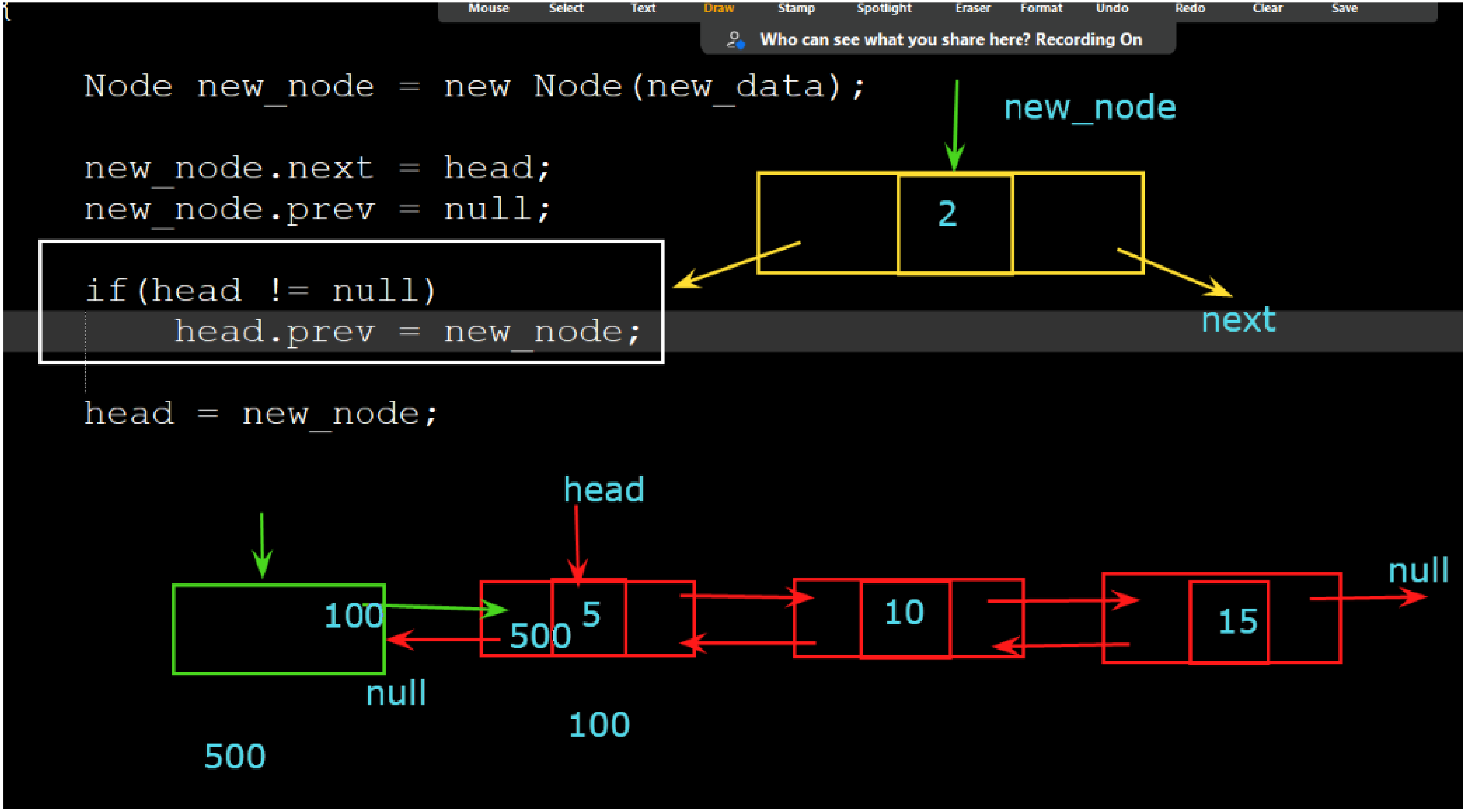
Doubly Linked List:



Who can see what you share here? Recording On

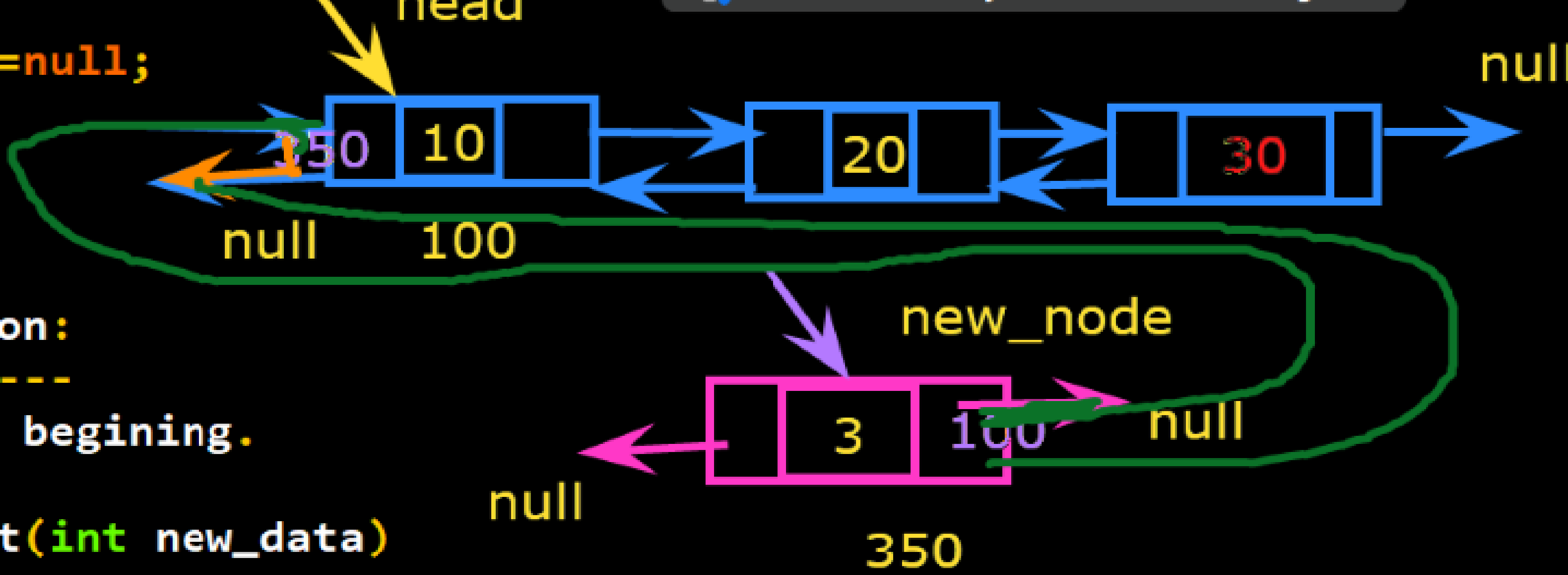
```
Node structure:
------------------

class Node{
    int data;
    Node prev;
    Node next;

    Node(int d)
    {
        data = d;
        prev=null;
        next=null;
    }
}
```





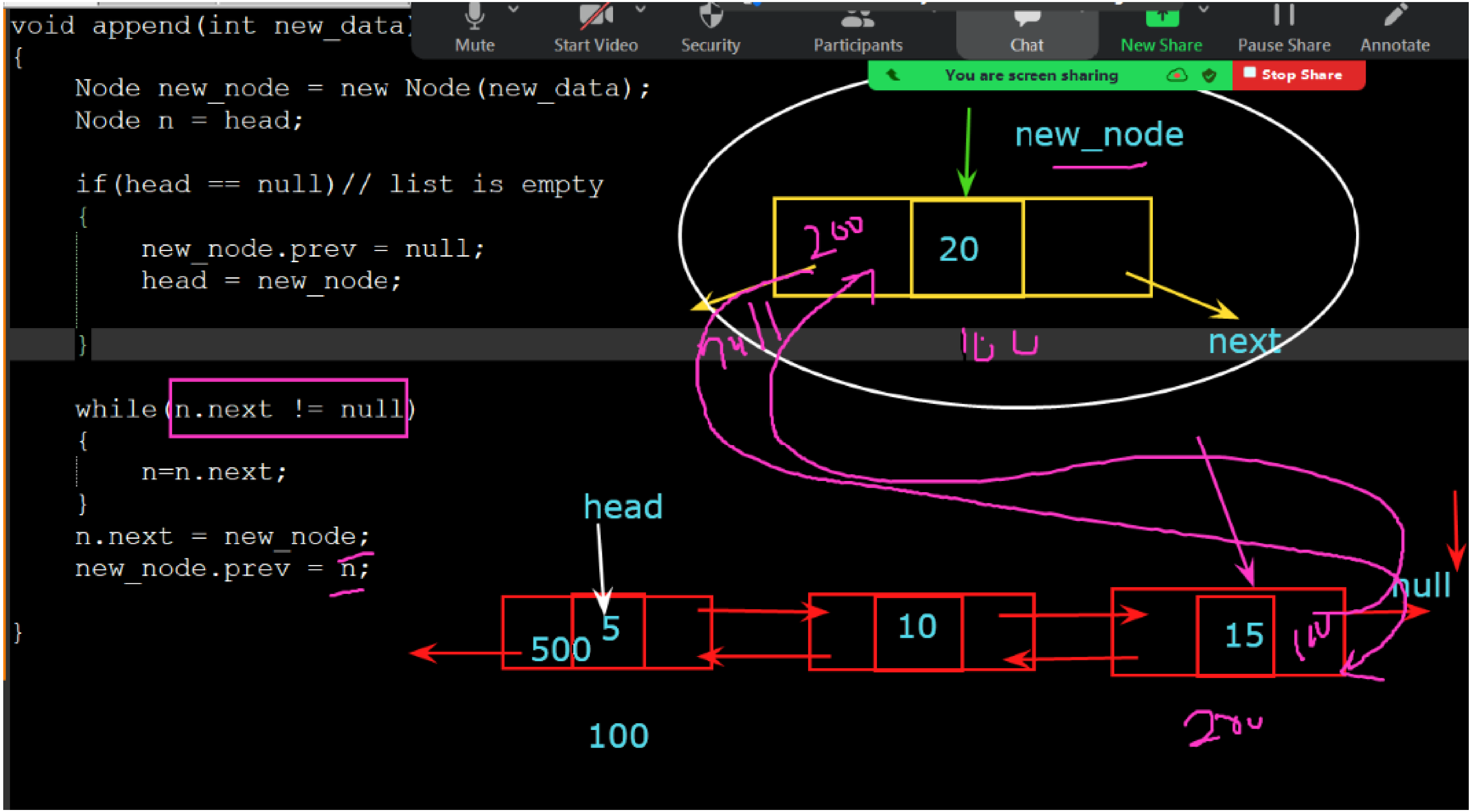
```
Node new_node = new Node(new_data);

new_node.next = head;
new_node.prev = null;

if(head != null)
    head.prev = new_node;

head = new_node;
```

```
        data = d;
        prev=next=null;
    }

}
```



Insertion Operation:
----
case 1: Insert at begining.

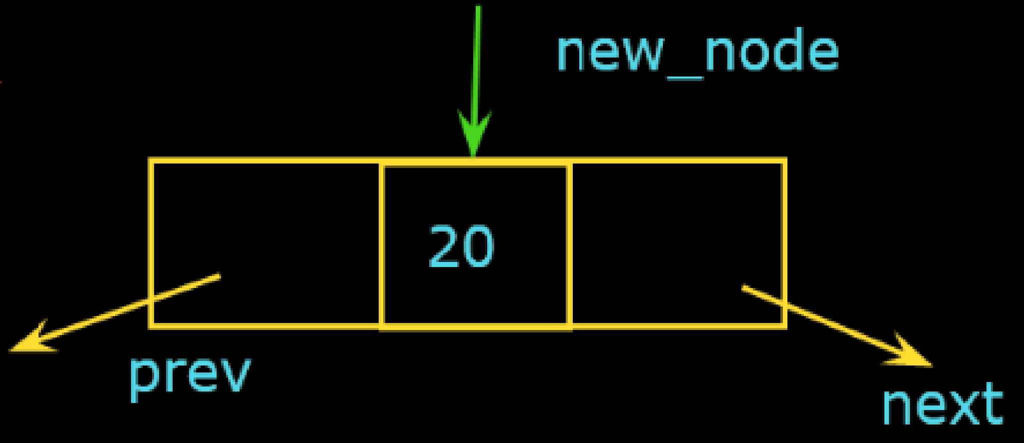
```
static void insert(int new_data)
{
    Node new_Node = new Node(new_data);
    new_Node.next = head;
    new_Node.prev = null;
    if(head != null)
        head.prev = new_node;
    head = new_Node;

}
```

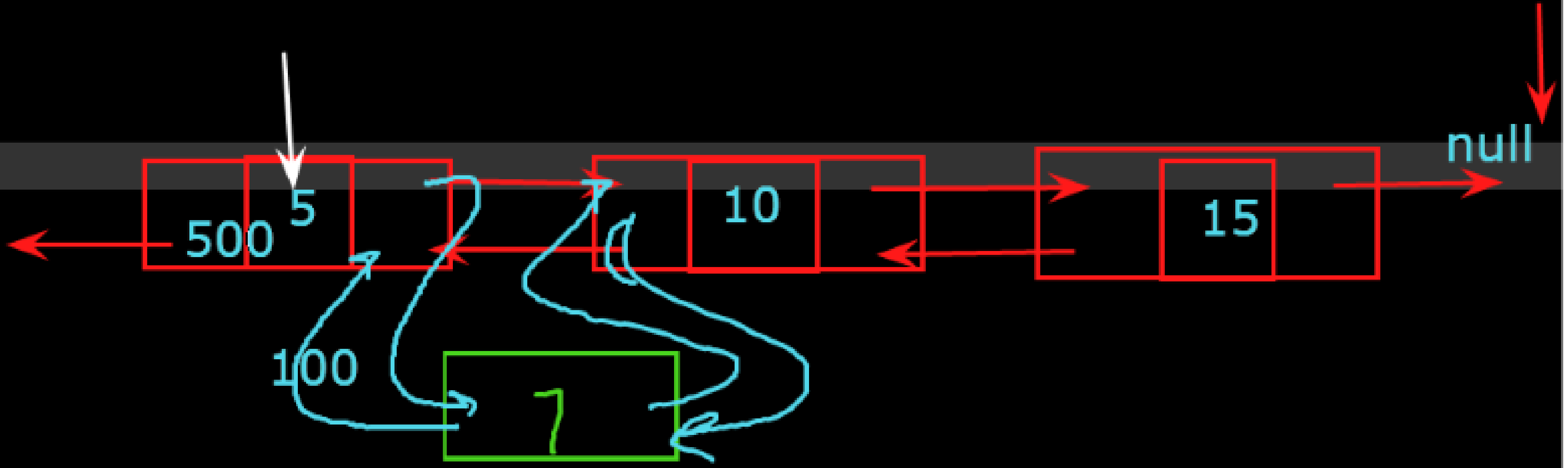


```
void append(int new_data)
{
    Node new_node = new Node(new_data);
    Node n = head;

    if(head == null)// list is empty
    {
        new_node.prev = null;
        head = new_node;

    }

    while(n.next != null)
    {
        n=n.next;
    }
    n.next = new_node;
    new_node.prev = n;

}
```

You are screen sharing Stop Share

```
case 3: Insertion in between two nodes.
-------------------------------------------
void insertAfter(Node n, int new_node)
{
    if(n == null)
        return;

    Node new_node = new Node(new_data);
    new_node.next = n.next;
    n.next.prev = new_node;
    new_node.prev = n;
    n.next = new_node;


}
```


new_node
20
prev
next


null
500
5
10
15
100
7