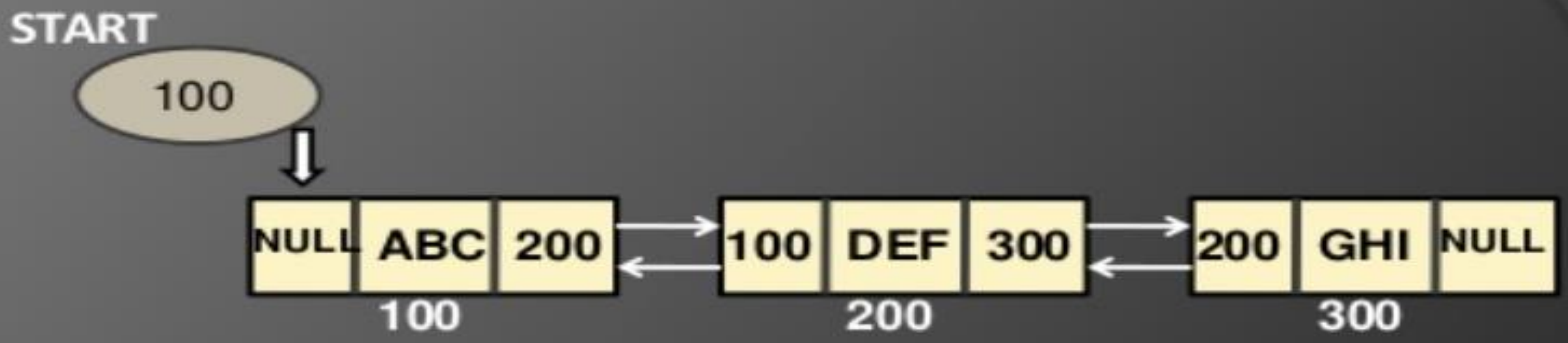
DOUBLY LINKED LIST

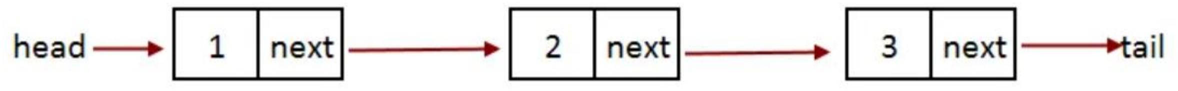


Doubly Linked List is a variation of Linked list in which navigation is possible in both ways, either forward and backward easily as compared to Single Linked List.

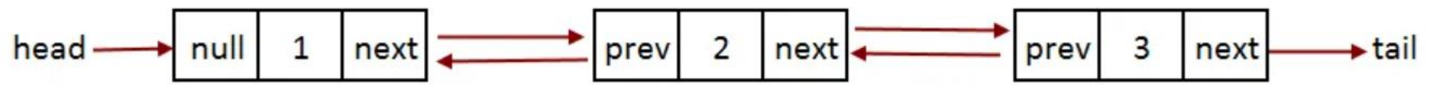
SS

Singly Linked List vs Doubly Linked List

Singly Linked List	Doubly Linked List
Easy Implement	Not easy
Less memory	More Memory
Can traverse only in forward direction	Traverse in both direction, back and froth



Singly Linked List



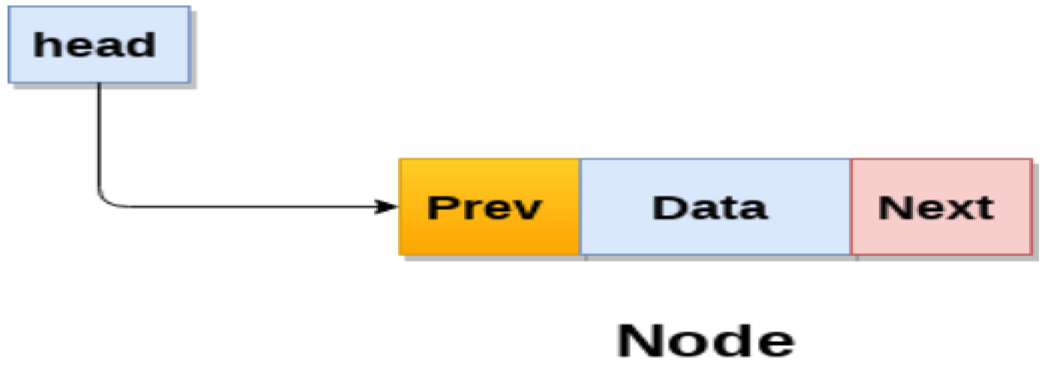
Doubly Linked List

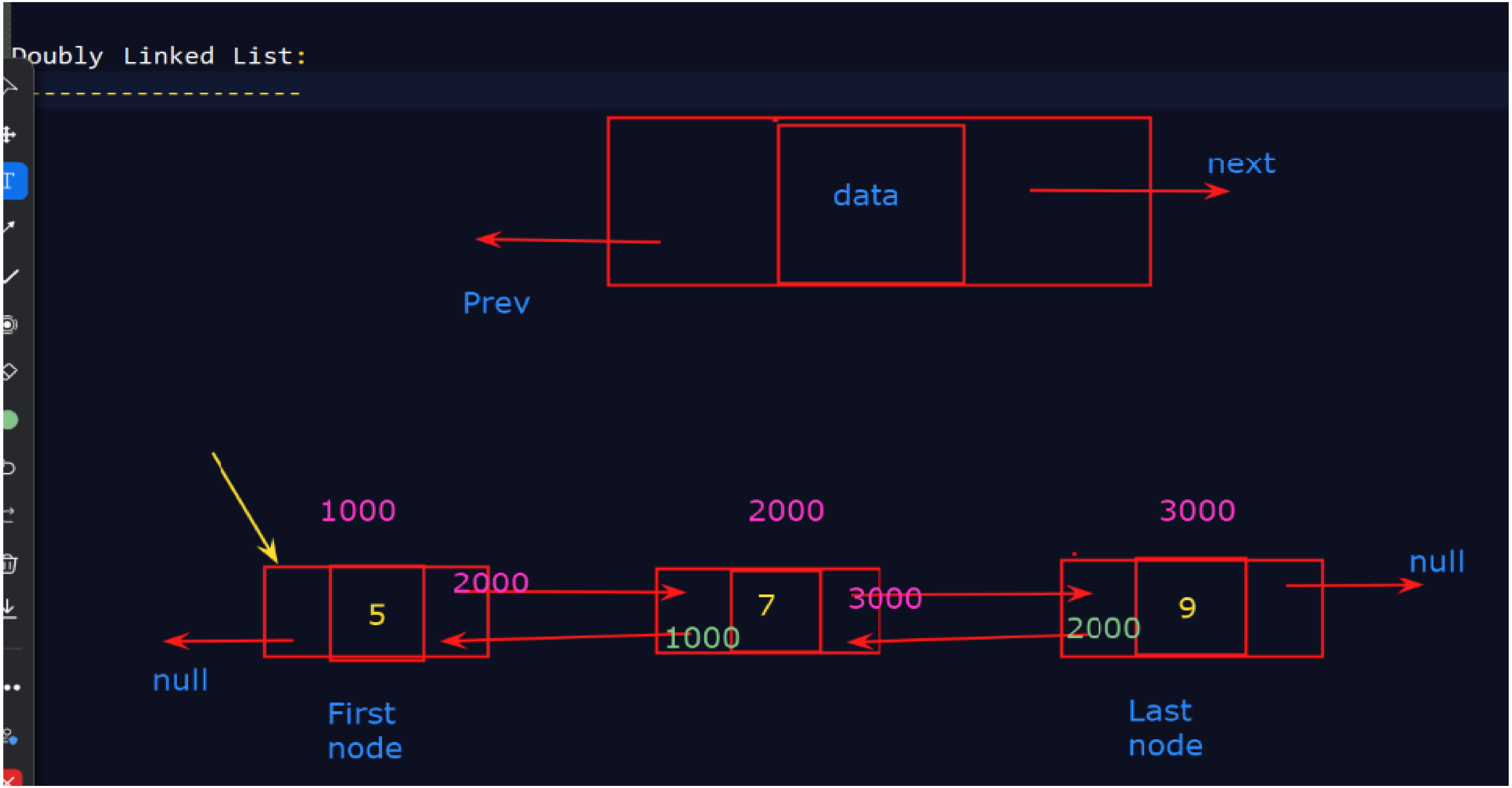
Doubly linked list

- Doubly linked list is a complex type of linked list
  - in which a node contains a pointer to the previous as well as the next node in the sequence.

- In a doubly linked list, a node consists of three parts:

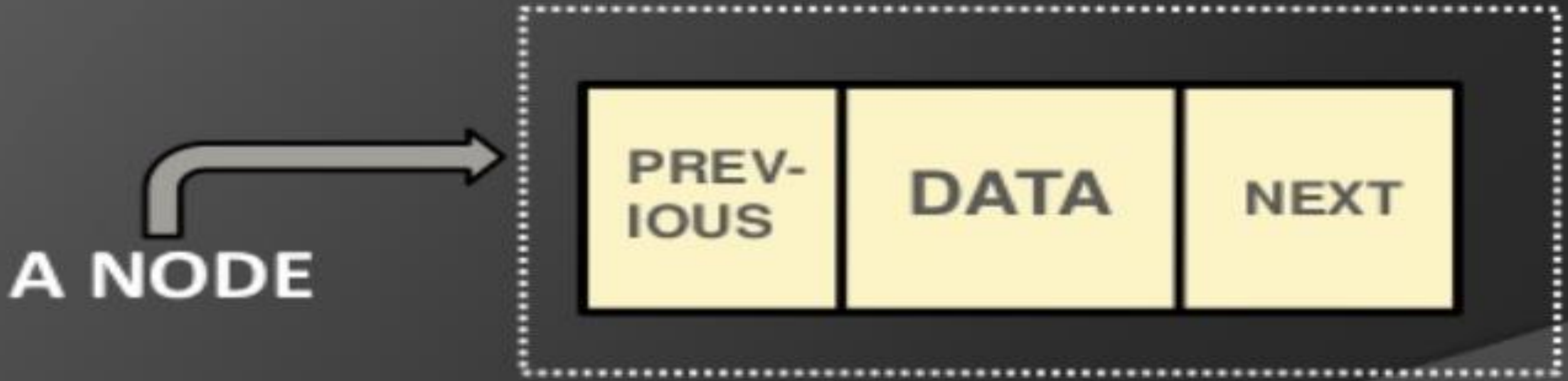
1. Data
2. Pointer to the previous node
3. pointer to the next node

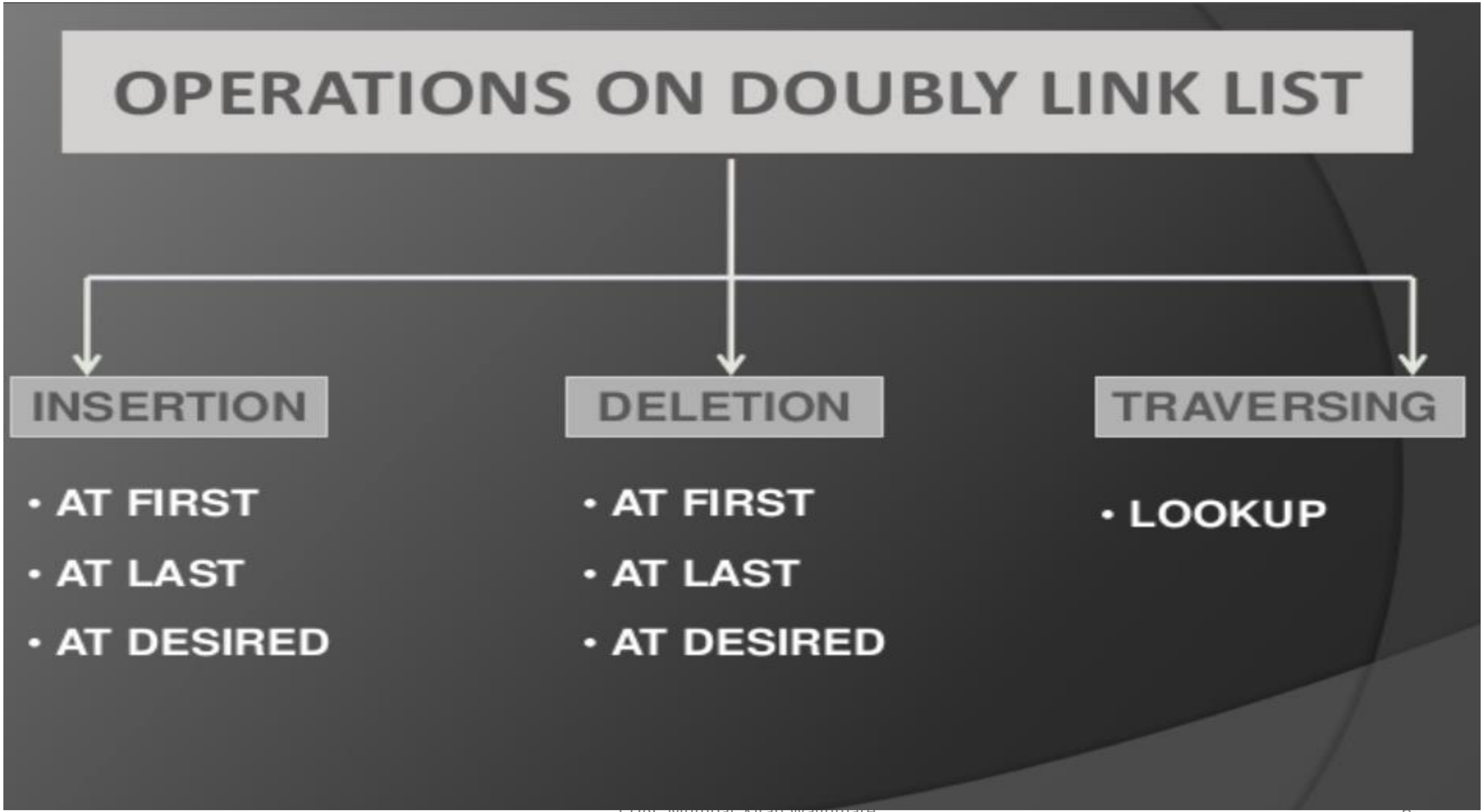


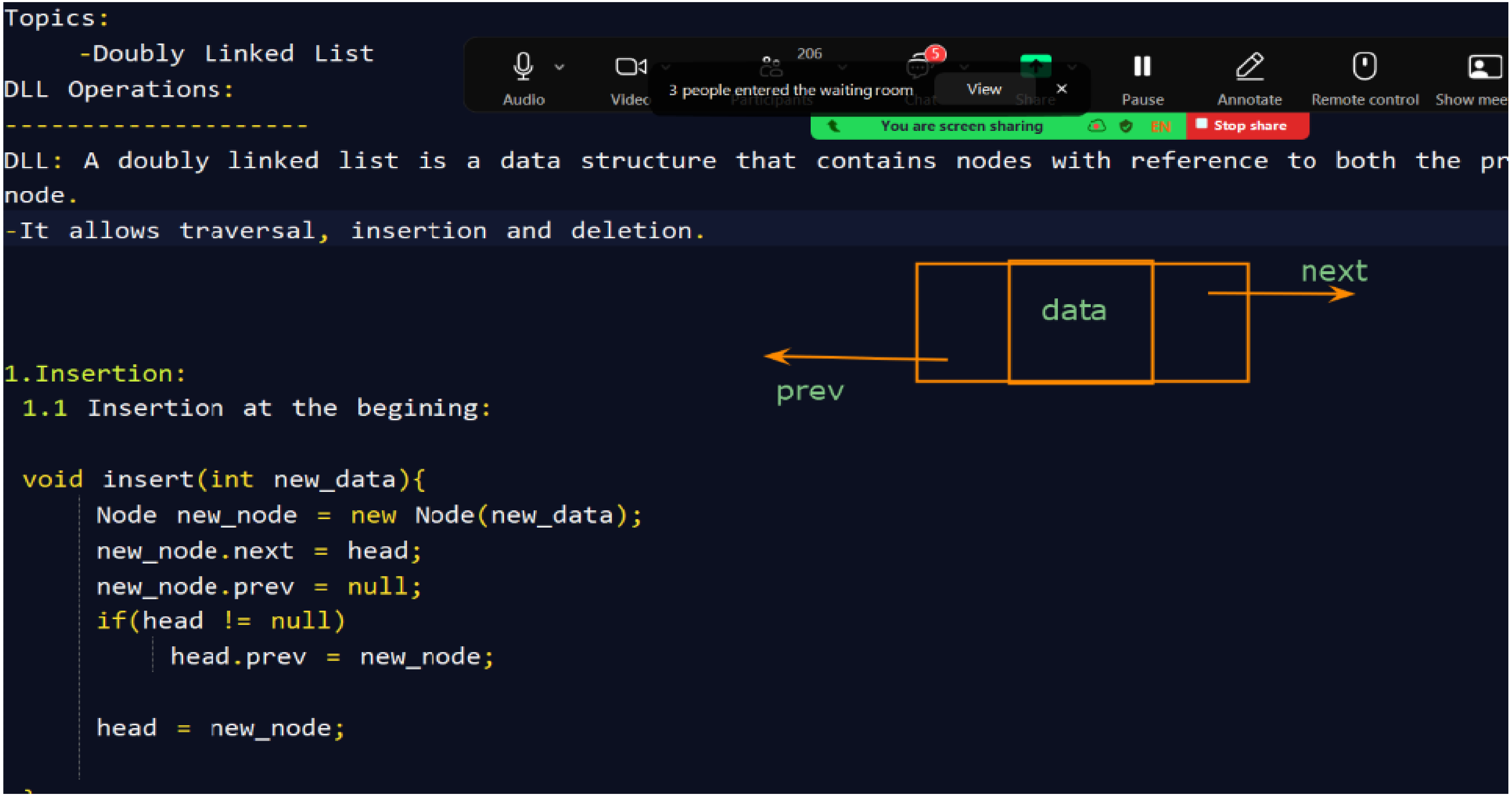

Doubly Linked List:

Why Doubly linked list ?

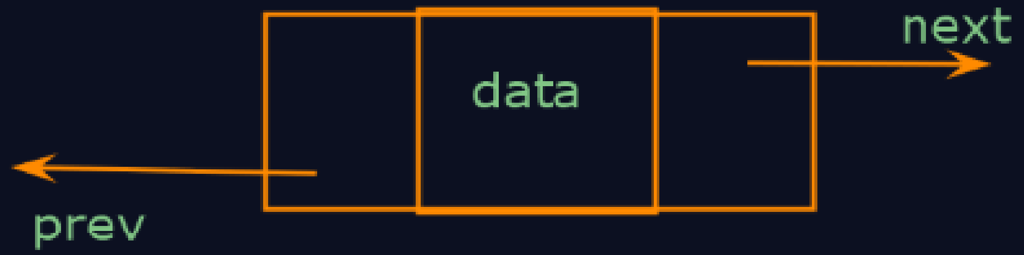
- In singly linked list we cannot traverse back to the previous node without an extra pointer. For ex to delete previous node.
- In doubly there is a link through which we can go back to previous node.



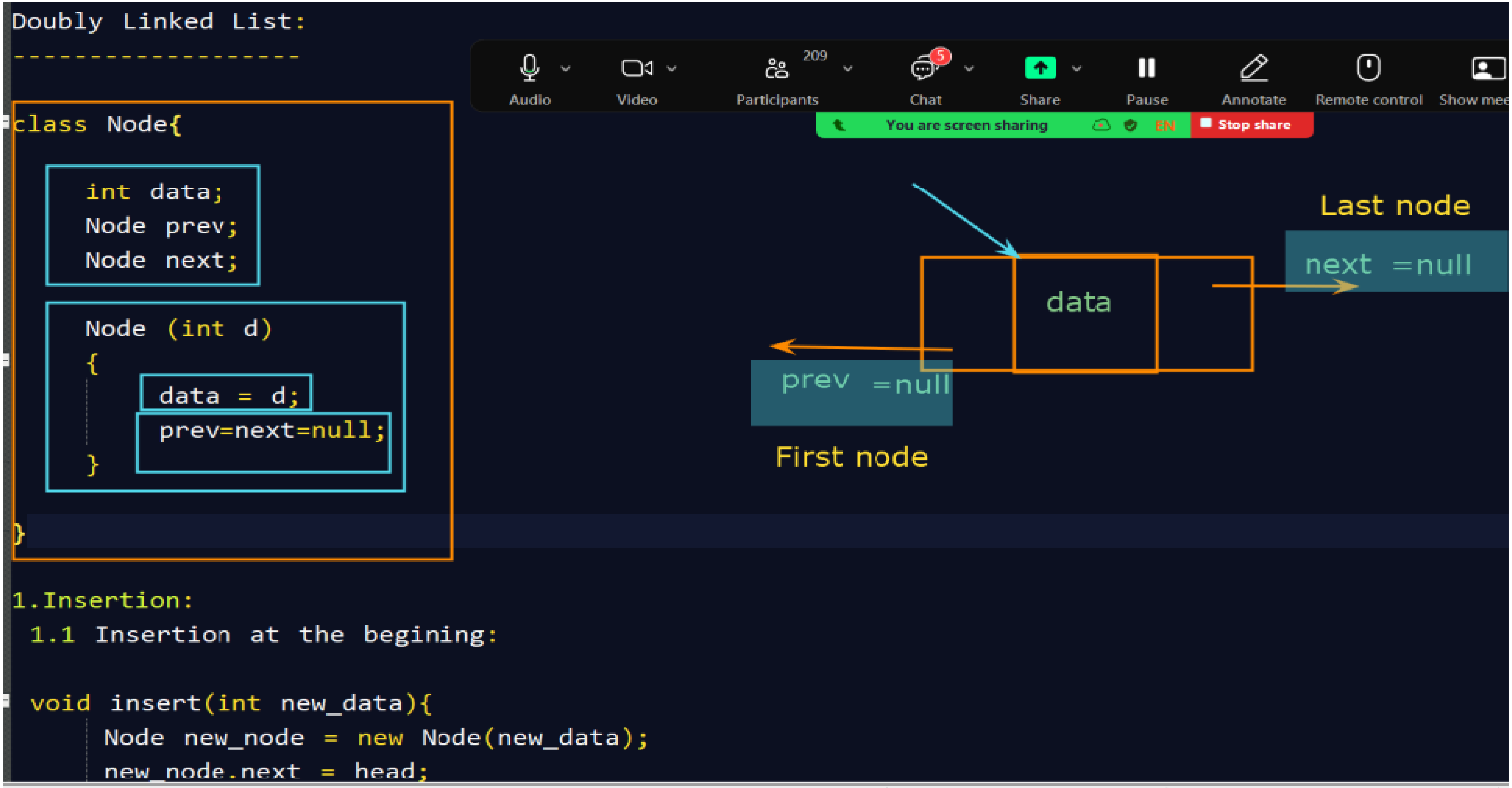

OPERATIONS ON DOUBLY LINK LIST
INSERTION
• AT FIRST
• AT LAST
• AT DESIRED
DELETION
• AT FIRST
• AT LAST
• AT DESIRED
TRAVERSING
• LOOKUP



```
Topics:
     -Doubly Linked List
DLL Operations:
--------------------
DLL: A doubly linked list is a data structure that contains nodes with reference to both the pr
node.
-It allows traversal, insertion and deletion.
```



```
1.Insertion:
 1.1 Insertion at the begining:

 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;

 }
```

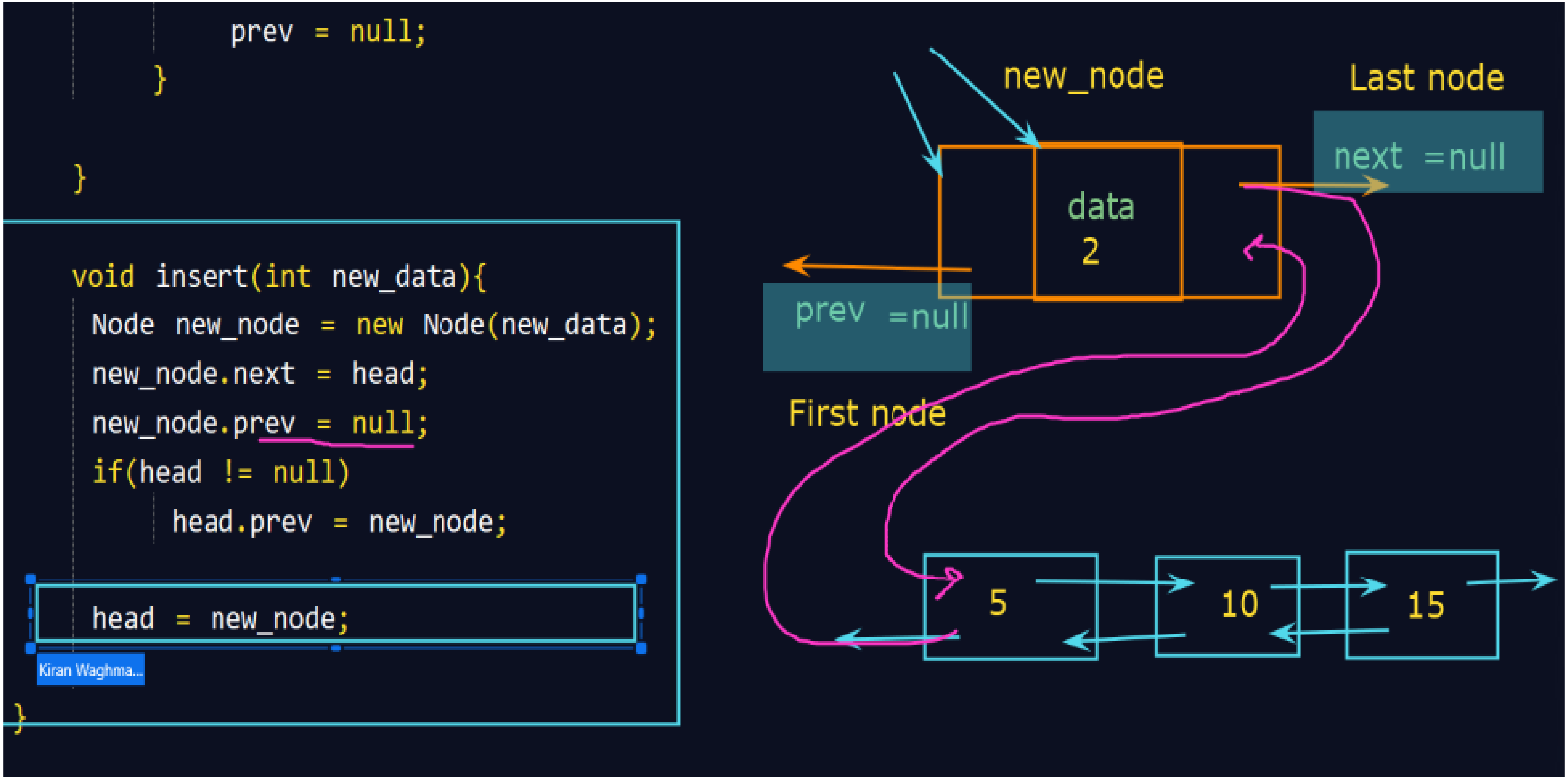


Doubly Linked List:

```
class Node{

    int data;
    Node prev;
    Node next;

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

}
```

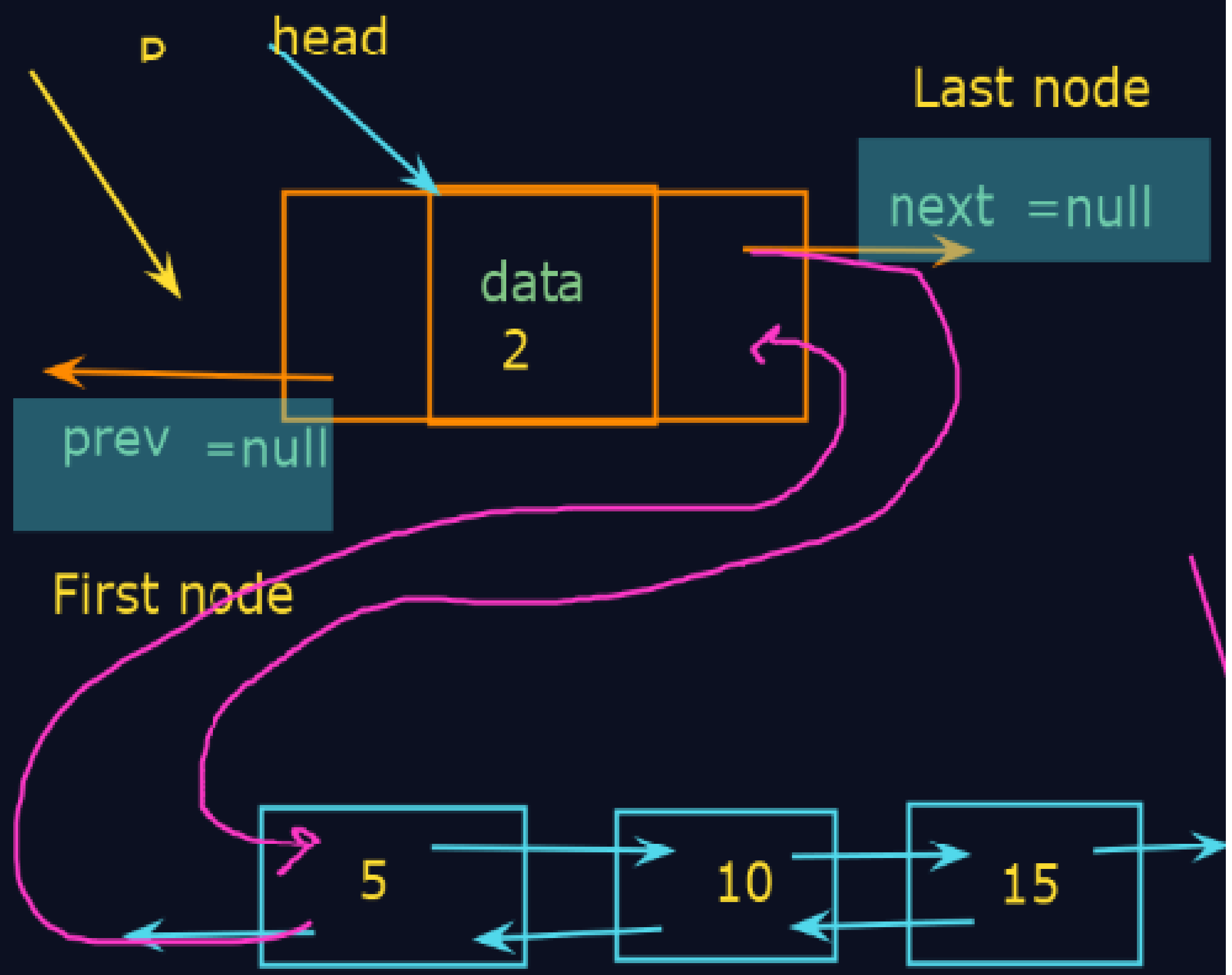
1.Insertion:
 1.1 Insertion at the begining:

```
 void insert(int new_data){
     Node new_node = new Node(new_data);
     new_node.next = head;
```



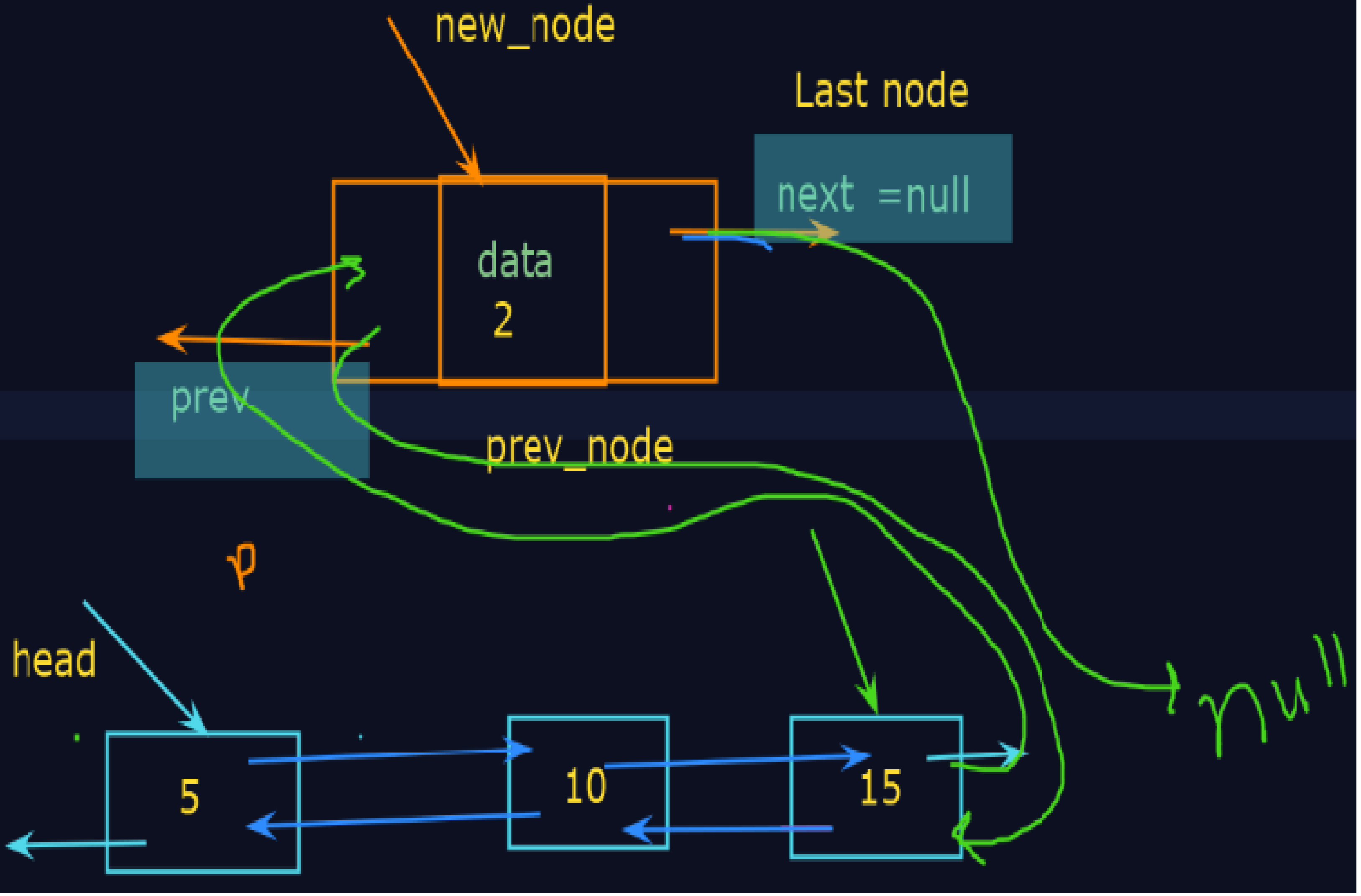
```
            prev = null;
        }

    }

    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;
}
```

```
public static void main(String args[])
{
    DLL1 d1 = new DLL1();
    d1.insert(6);
    d1.insert(7);
    d1.insert(8);
    d1.insert(9);
    d1.display(d1.head);
    System.out.println();
}
```

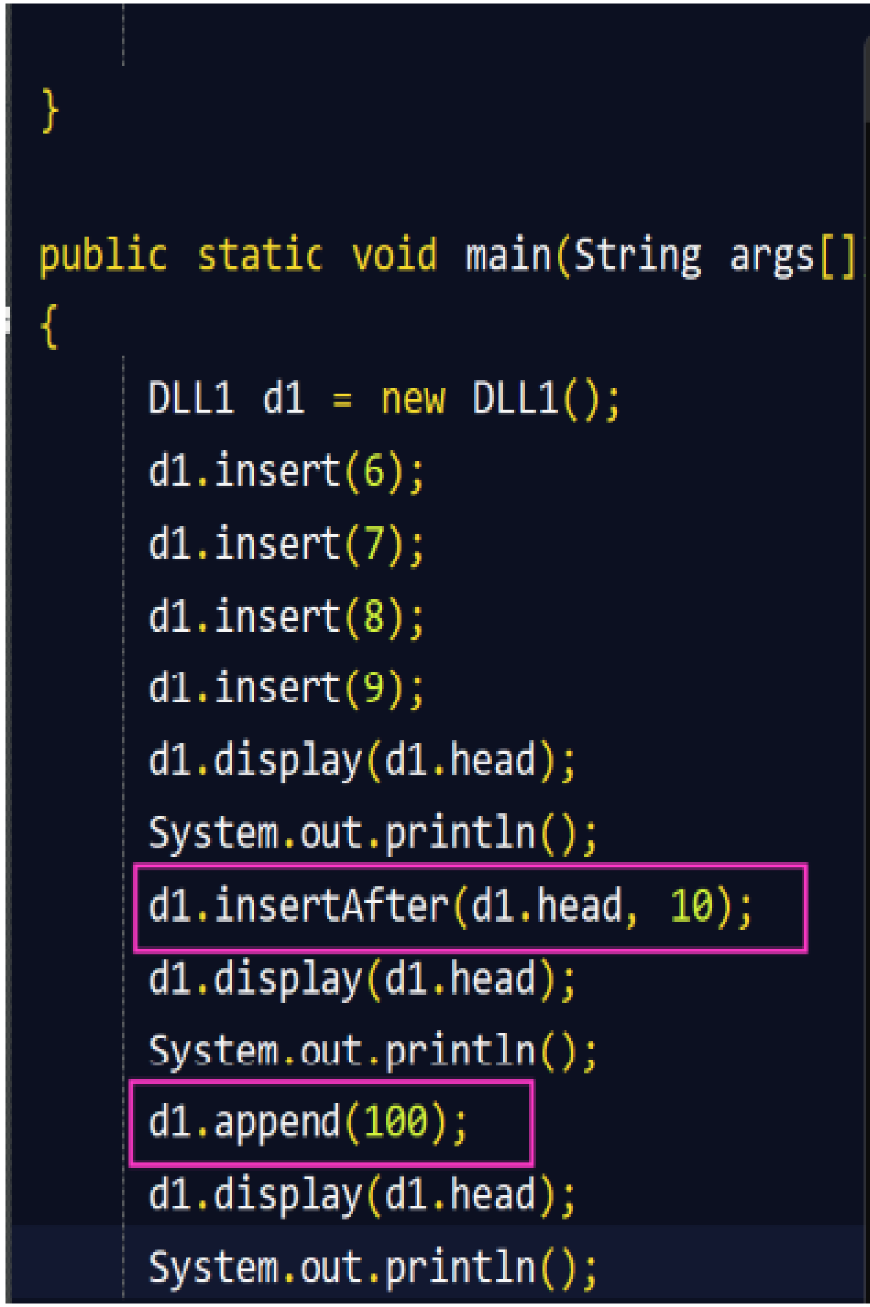


1.2 Insertion in between

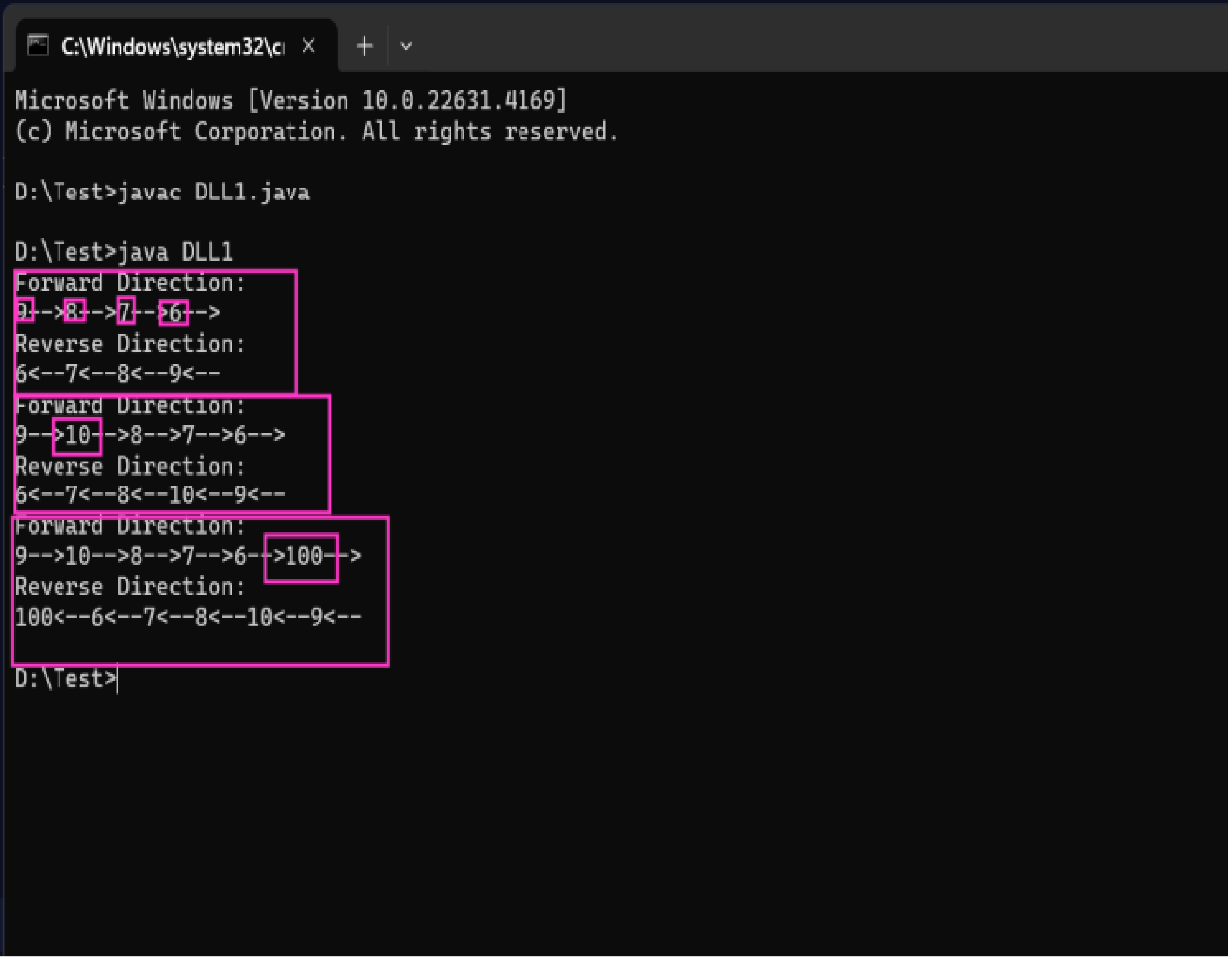
```
void insertAfter(Node prev_node, int new_data)
{
    if(prev_node == null)
    {
        System.out.println("Node cannot exist!");
        return;
    }

    Node new_node = new Node(new_data);
    new_node.next = prev_node.next;
    prev_node.next = new_node;
    new_node.prev =prev_node;
    if(new_node.next != null)
        new_node.next.prev =new_node;

}
```



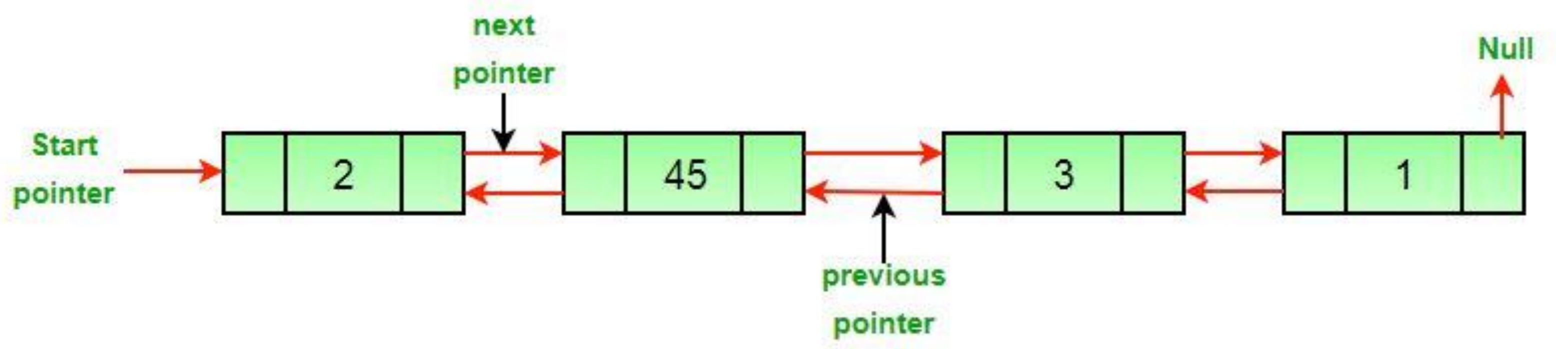
1.3 Insertion at the end

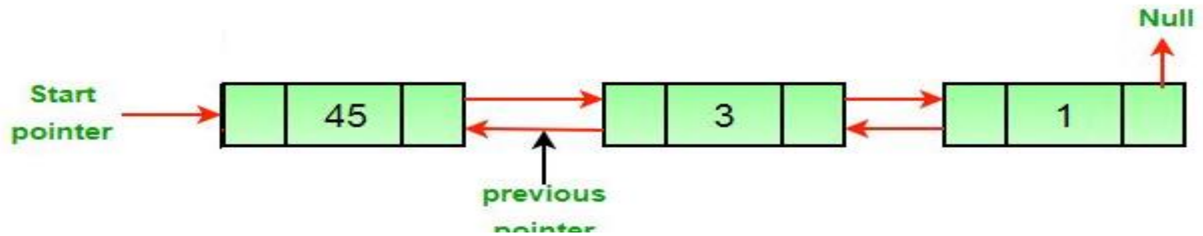


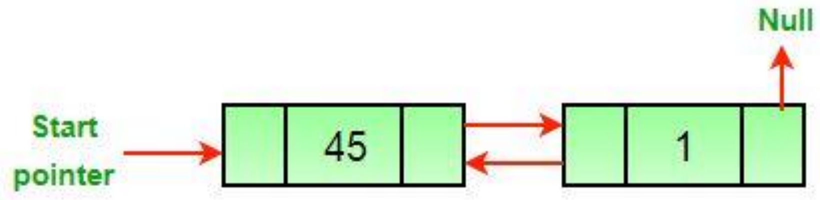
```
}

public static void main(String args[]
{
    DLL1 d1 = new DLL1();
    d1.insert(6);
    d1.insert(7);
    d1.insert(8);
    d1.insert(9);
    d1.display(d1.head);
    System.out.println();
    d1.insertAfter(d1.head, 10);
    d1.display(d1.head);
    System.out.println();
    d1.append(100);
    d1.display(d1.head);
    System.out.println();
```

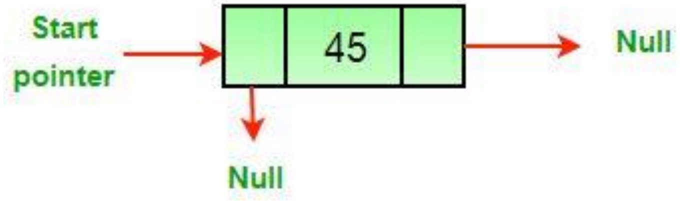


```
Microsoft Windows [Version 10.0.22631.4169]
(c) Microsoft Corporation. All rights reserved.

D:\Test>javac DLL1.java

D:\Test>java DLL1
Forward Direction:
9-->8-->7-->6-->
Reverse Direction:
6<--7<--8<--9<--
Forward Direction:
9-->10-->8-->7-->6-->
Reverse Direction:
6<--7<--8<--10<--9<--
Forward Direction:
9-->10-->8-->7-->6-->100-->
Reverse Direction:
100<--6<--7<--8<--10<--9<--

D:\Test>
```


next pointer
Null
Start pointer
2
45
3
1
previous pointer


Null
Start pointer
45
3
1
previous


Null
Start pointer
45
1


Start pointer
45
Null
Null

Thanks