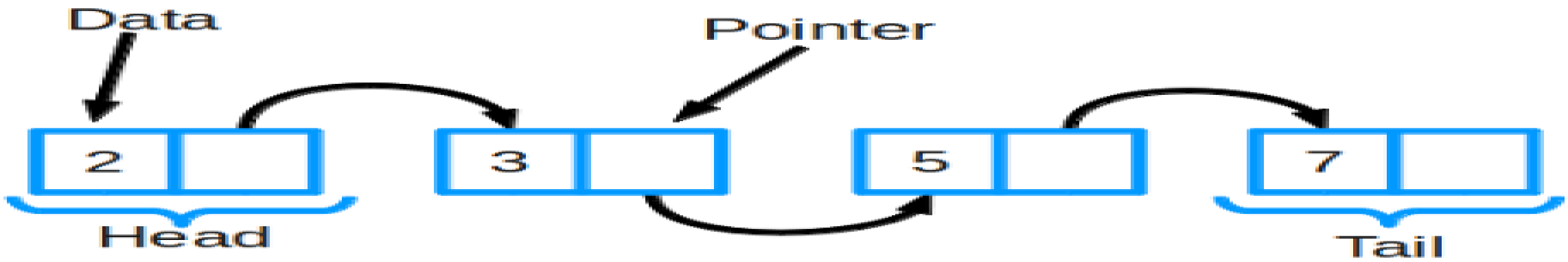
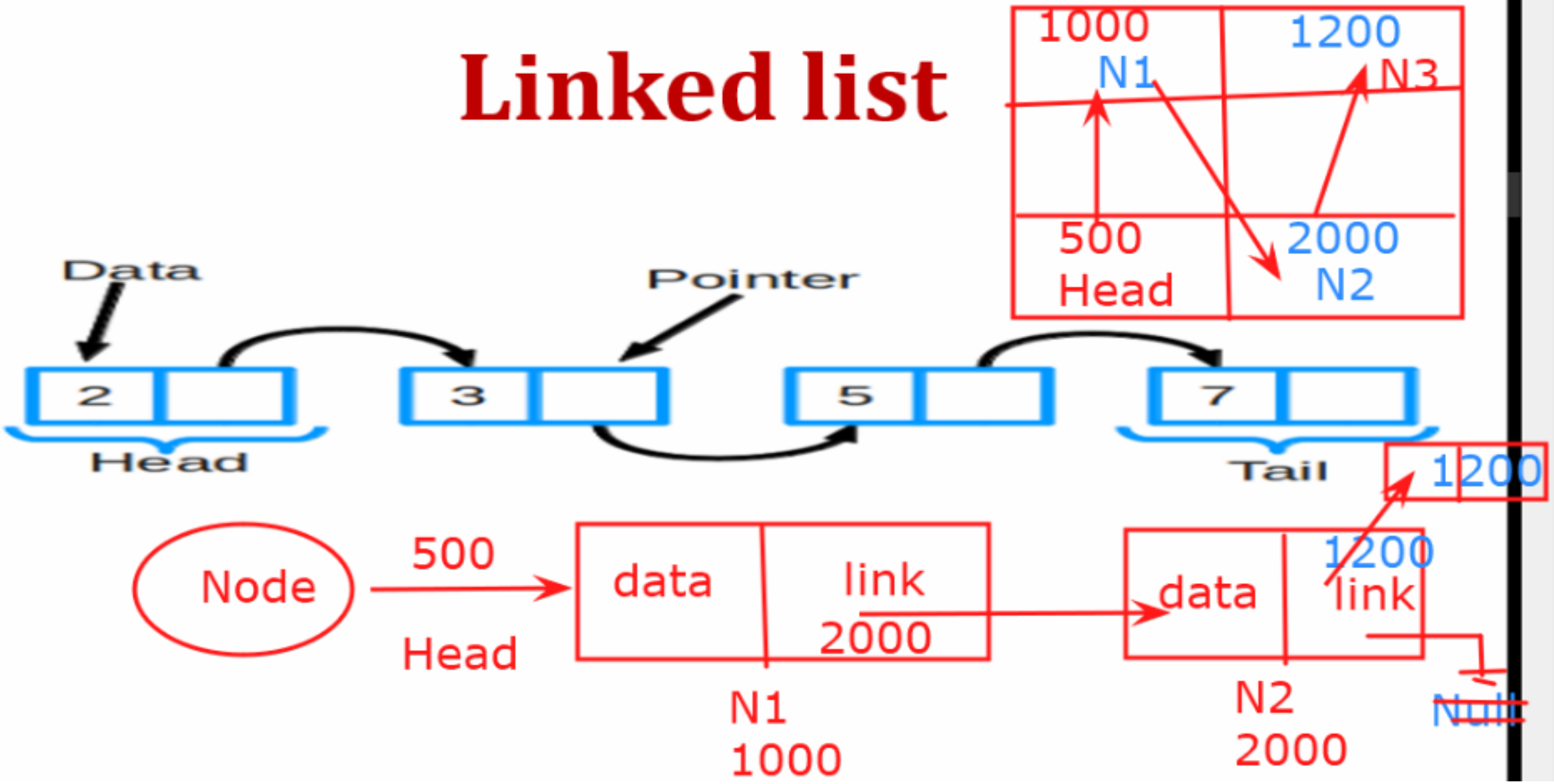
Algorithms & Data Structure

Kiran Waghmare

Linked list

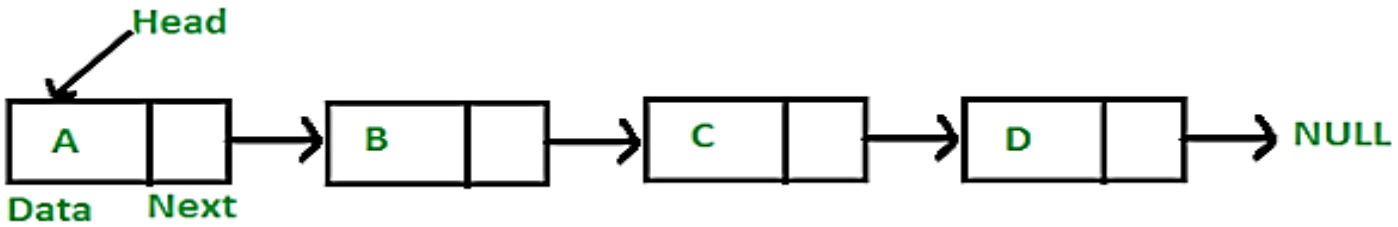


Linked list


1000
N1
1200
N3
500
Head
2000
N2
Data
Pointer
2
3
5
7
Head
Tail
1200
Node
500
Head
data
link
2000
N1
1000
data
1200
link
N2
2000
Null

Linked List Representation

- **Linked list can be visualized as a chain of nodes, where every node points to the next node.**



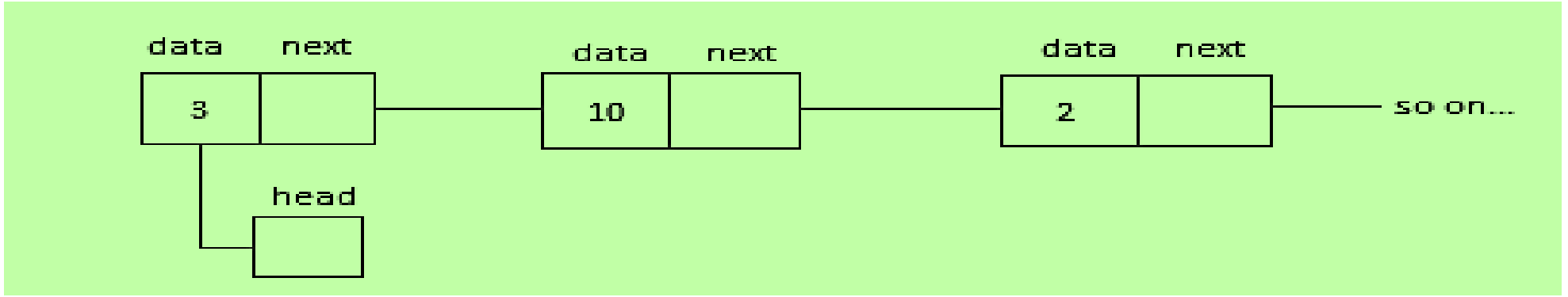
- **As per the above illustration, following are the important points to be considered.**

1. Linked List contains a **link element** called **first.**
2. Each link carries a **data field(s)** and a **link field** called **next.**
3. Each link is **linked with its next link** using its **next link.**
4. **Last link carries a link as null** to mark the end of the list.

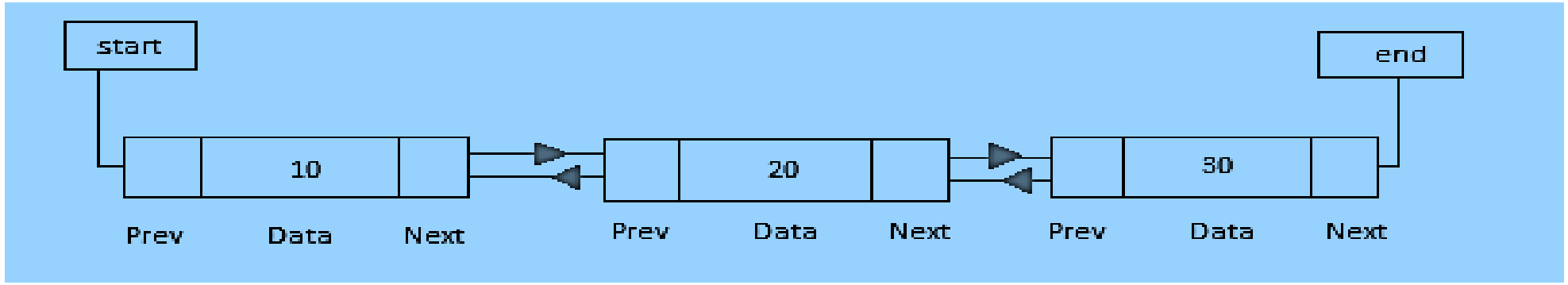
Types of Linked List

- **Following are the various types of linked list.**
  1. **Simple Linked List** – Item navigation is forward only.
  2. **Doubly Linked List** – Items can be navigated forward and backward.
  3. **Circular Linked List** – Last item contains link of the first element as next and the first element has a link to the last element as previous.

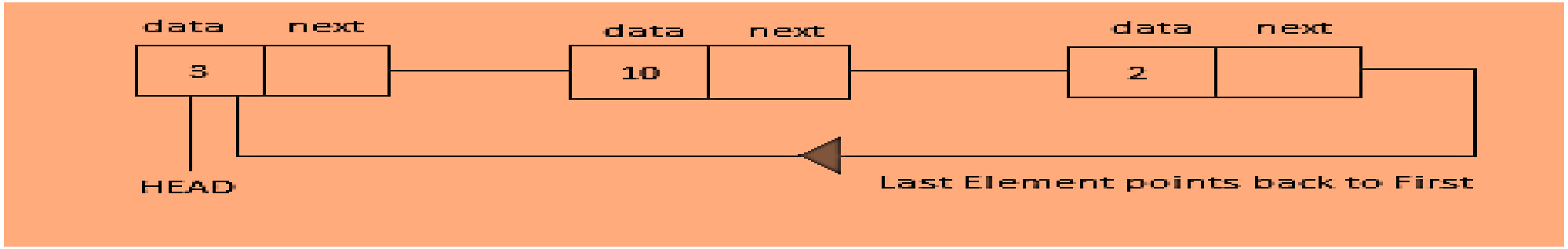
- **Simple Linked List**



- **Doubly Linked List**

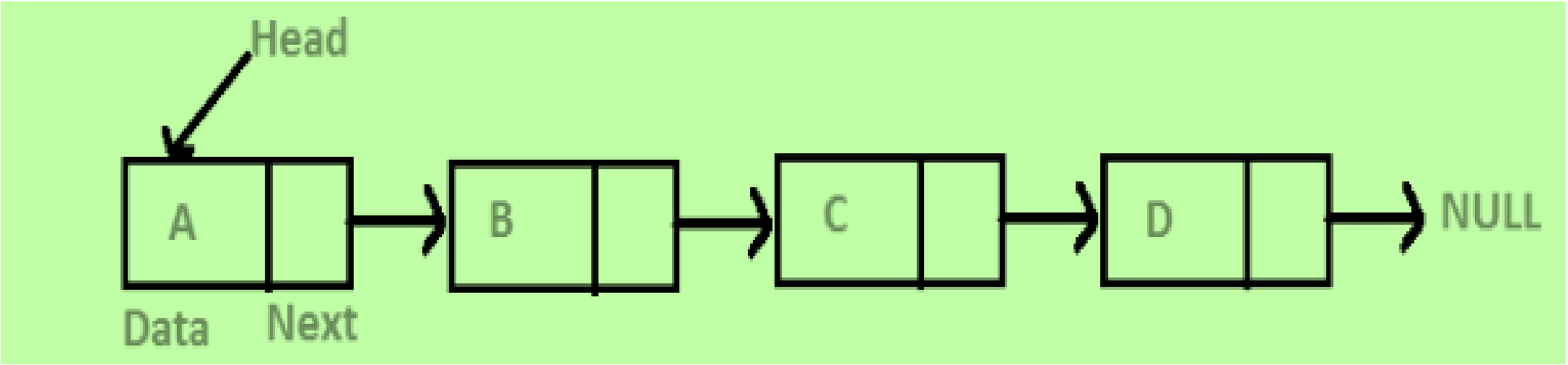


- **Circular Linked List**



Singly Linked List

- **Singly Linked Operations: Insert, Delete, Traverse, search, Sort, Merge**



Applications of Linked Lists

1. Linked lists are used to implement stacks, queues, graphs, etc.
2. Linked lists let you insert elements at the beginning and end of the list.
3. In Linked Lists we don't need to know the size in advance.

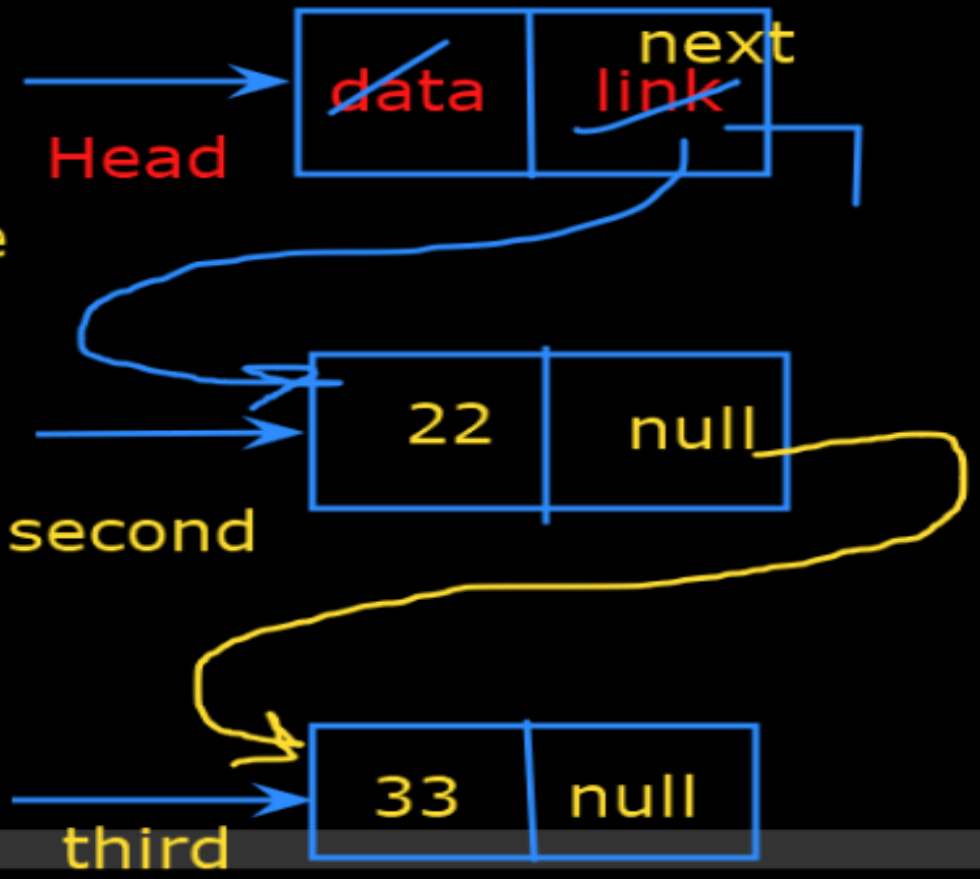
Basic Operations

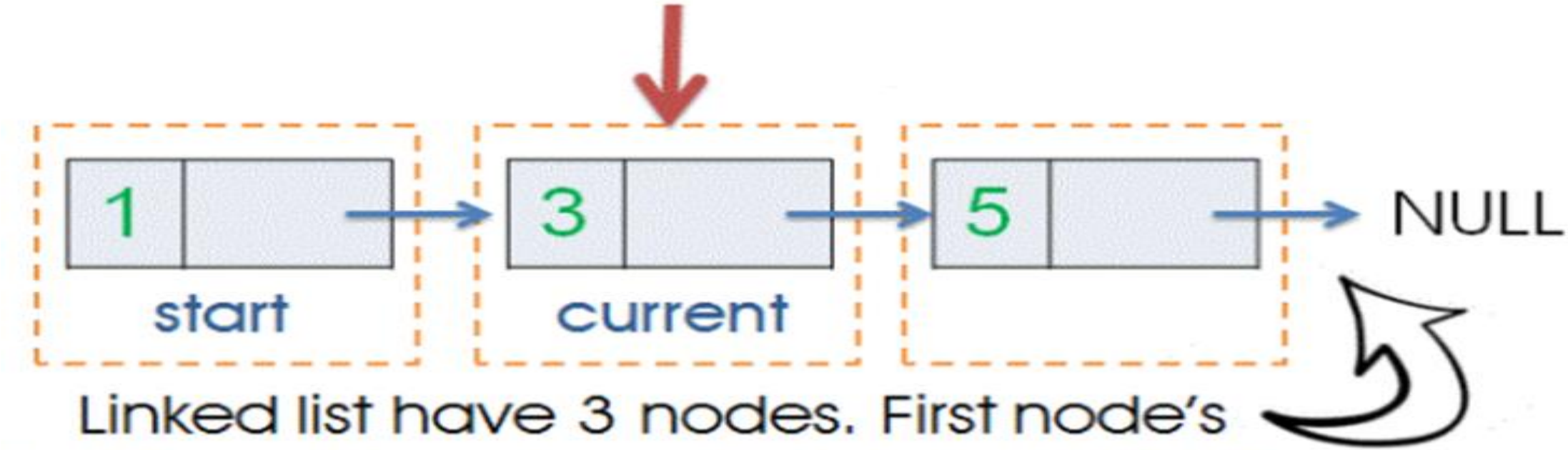
- **Following are the basic operations supported by a list.**
  1. **Insertion** − Adds an element at the beginning of the list.
  2. **Deletion** − Deletes an element at the beginning of the list.
  3. **Display** − Displays the complete list.
  4. **Search** − Searches an element using the given key.
  5. **Delete** − Deletes an element using the given key.

```
class List1
{
Node head;//Start of list

class Node
{
    int data;
    Node next;

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

public static void main(String args[])
{
    List1 ll = new List();

    ll.head = new Node(11);
    Node second = new Node(22);
    Node third = new Node(33);

    ll.head.next = second;
    second.next = third;


}

}
```

=>NODE

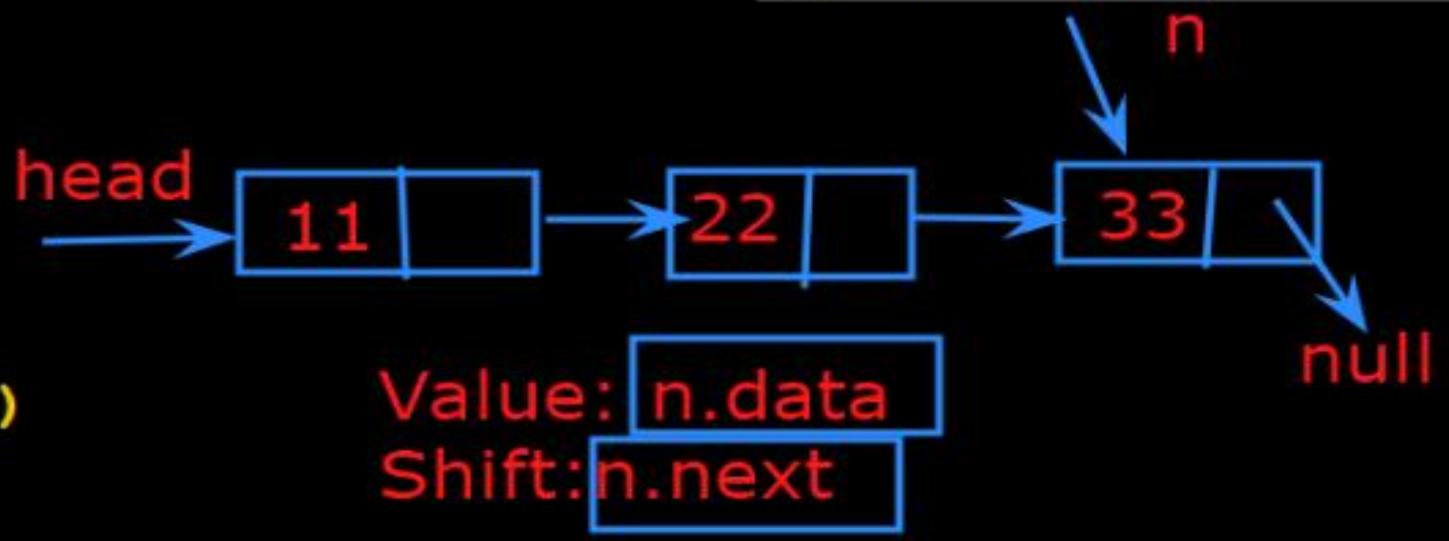
=>setting value in node





Linked list have 3 nodes. First node's Address will be stored in Pointer Variable "**start**" of type node.

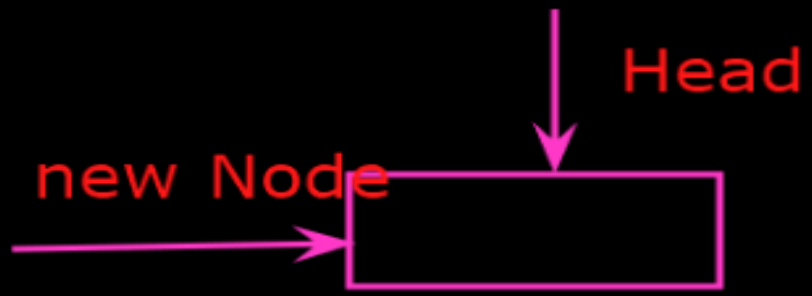
```
    Node(int d)
    {
        data = d;
        next = null;
    }

}
public void display()
{
    Node n = head;
    while(n != null)
    {
        System.out.print(n.data+ "--->");
        n = n.next;

    }

}
```



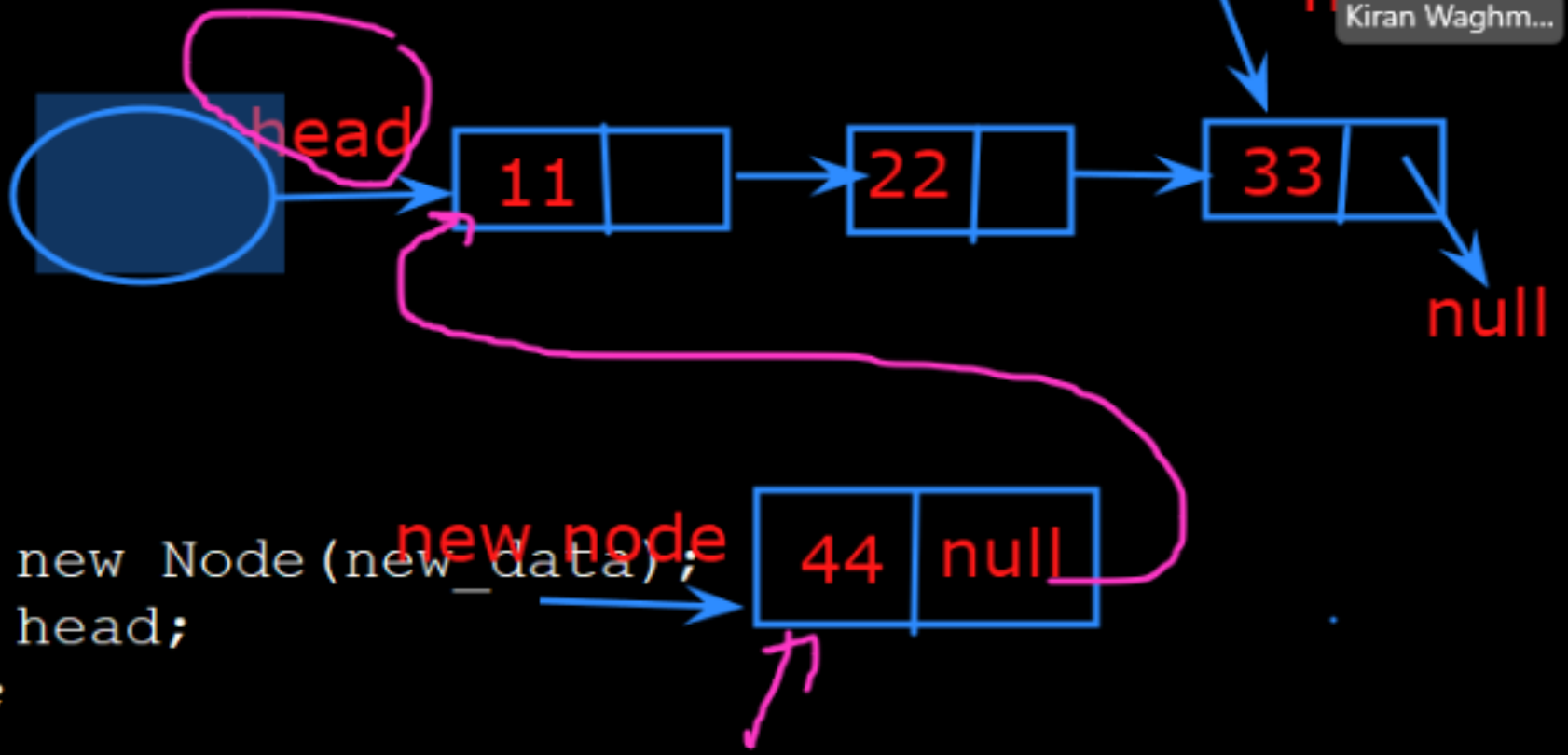
```
----------------------------
Insertion in Linked List
```

Case 1:

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

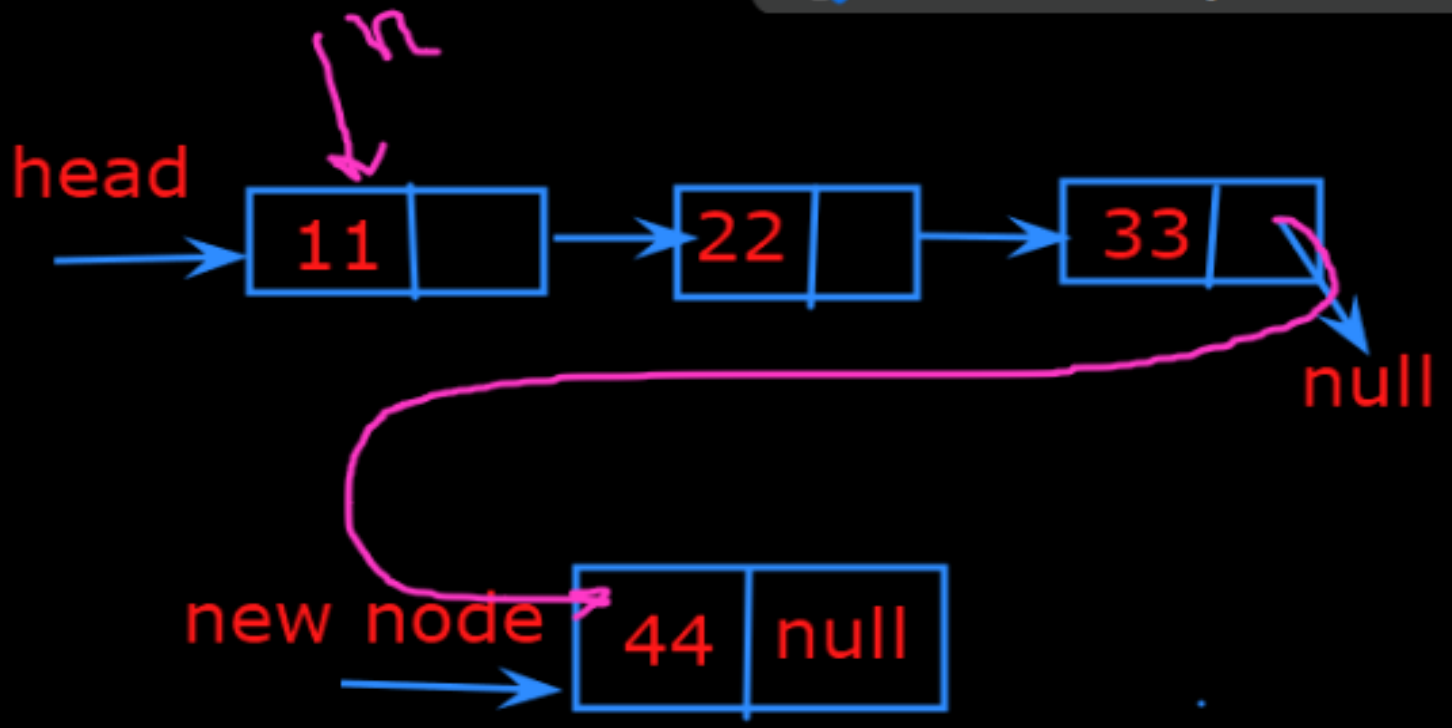


case 2:

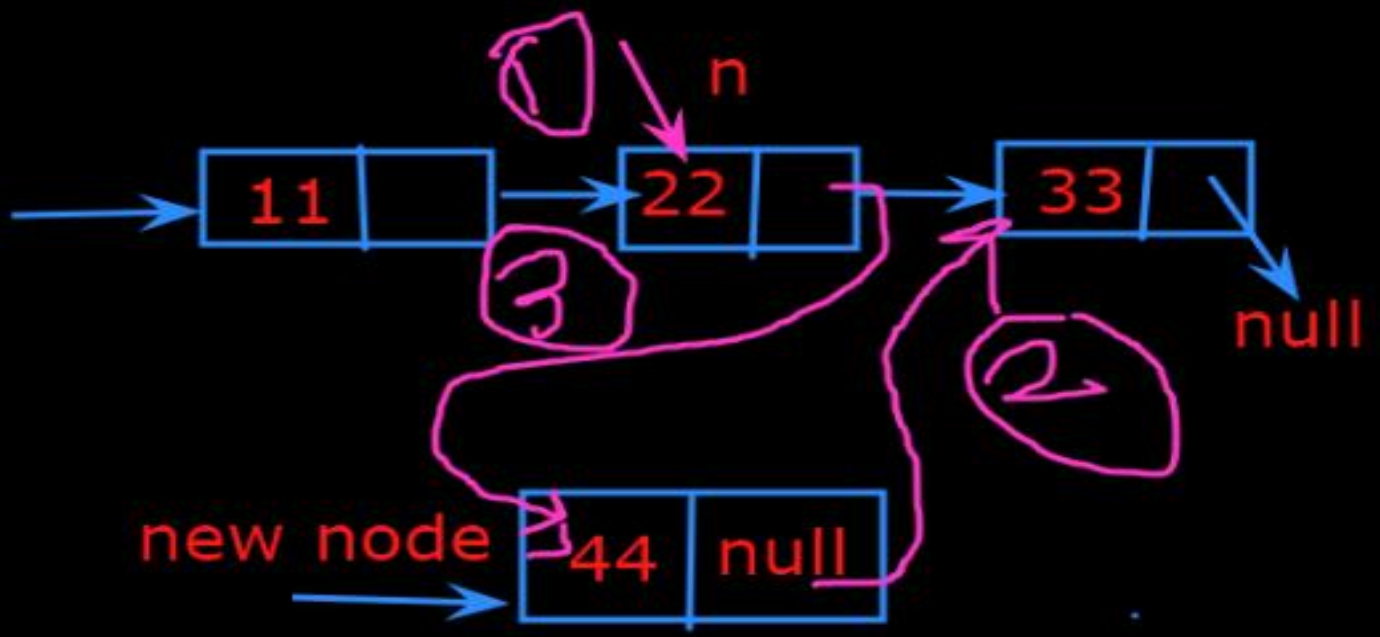



```
Node new_Node = new Node(new_data);
new_Node.next = head;
head = new_Node;
```

```
case 3:



Node head;

if (head = null)

else
while(n.next !=null)
{
    n=n.next;
}
n.next = new_Node;
```

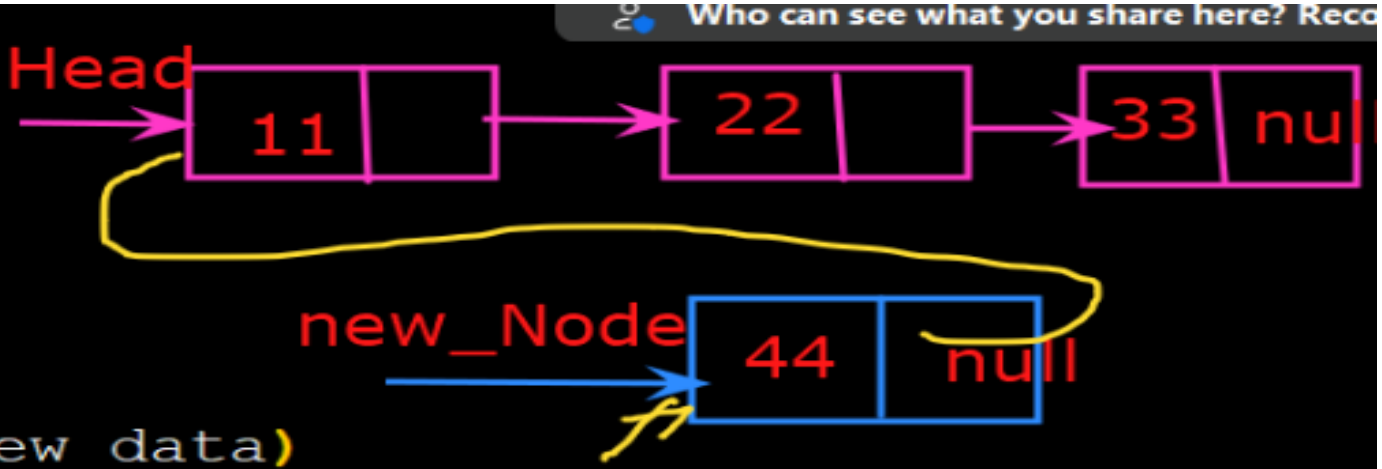


```
case 4:
```

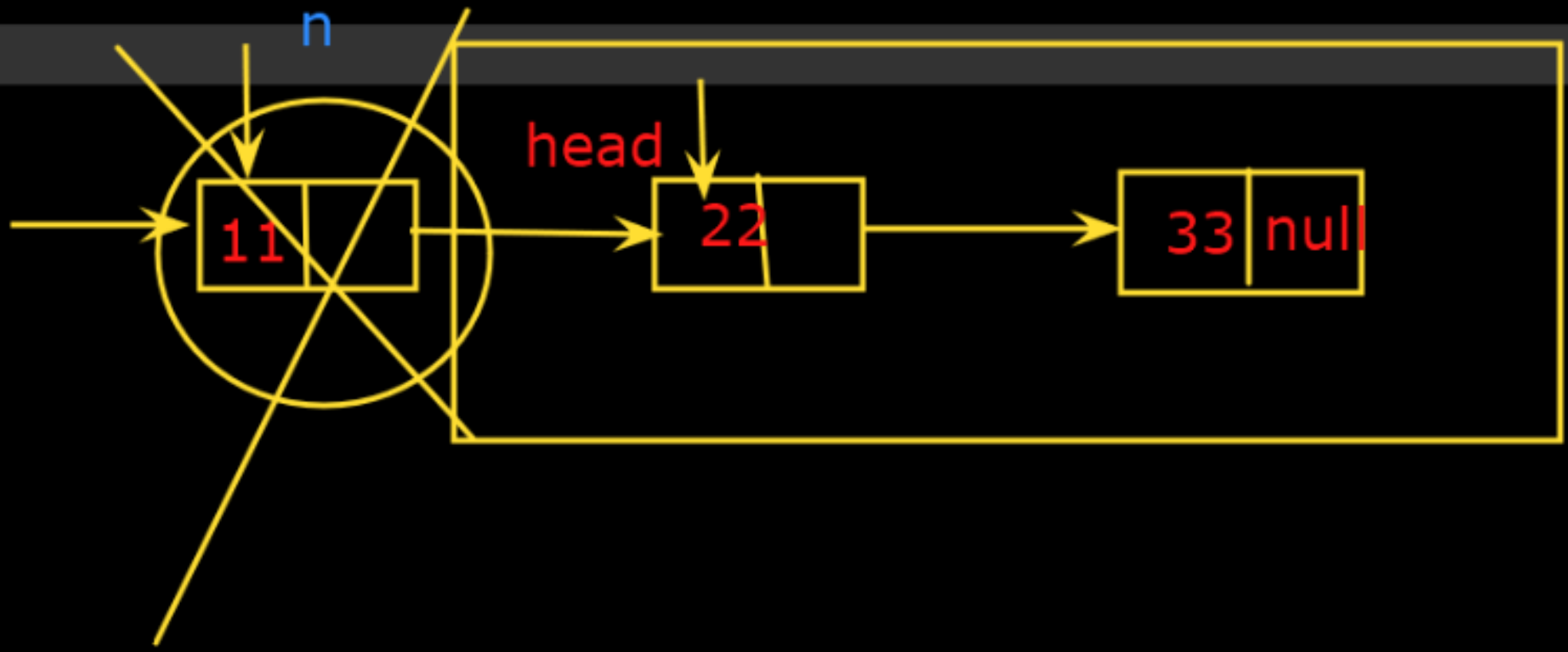



```
Node new_Node = new Node(new_data);
n=n.next;
new_Node.next =n.next();
n.next = new_Node;
```

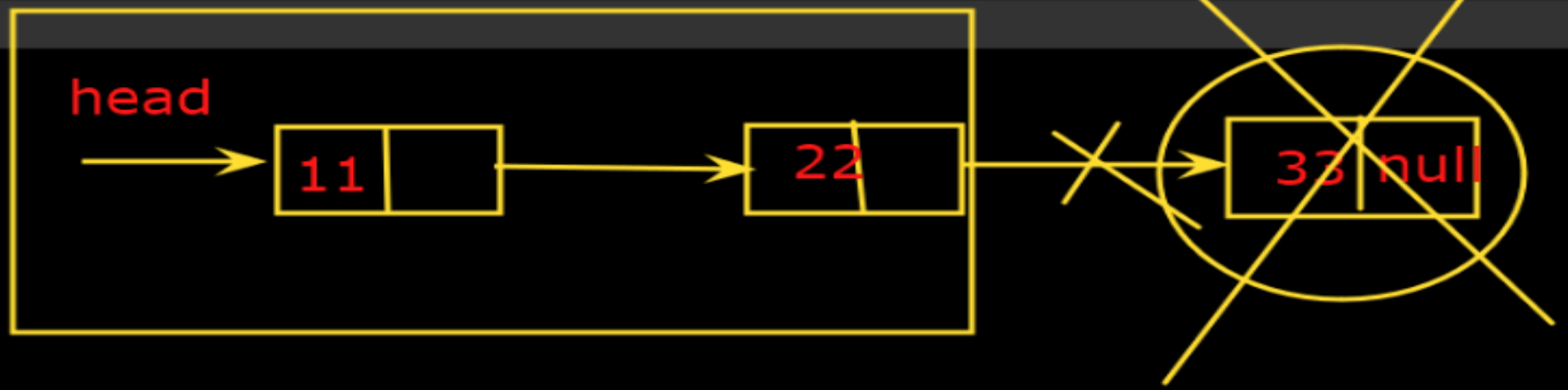
```
            n = n.next;

        }

    }

    //Insert at begining
     public void insert(int new_data)
     {
        Node new_Node = new Node(int new_data);.
        new_Node.next = head;
        head = new_Node;

     }

public static void main(String args[])
{
        List3 l1 = new List3();

        l1.head = new Node(11);
        Node second  = new Node(22);
```

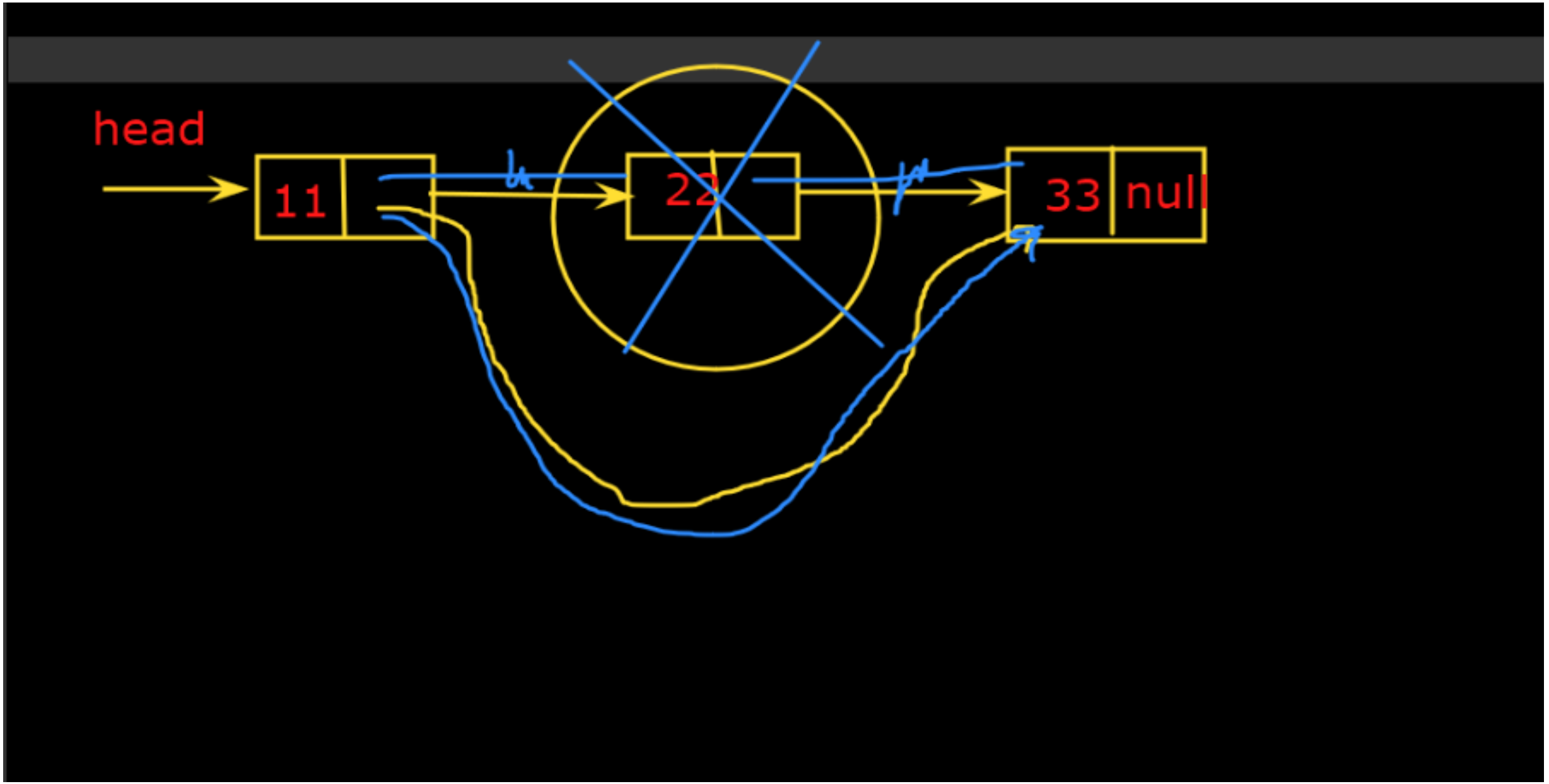


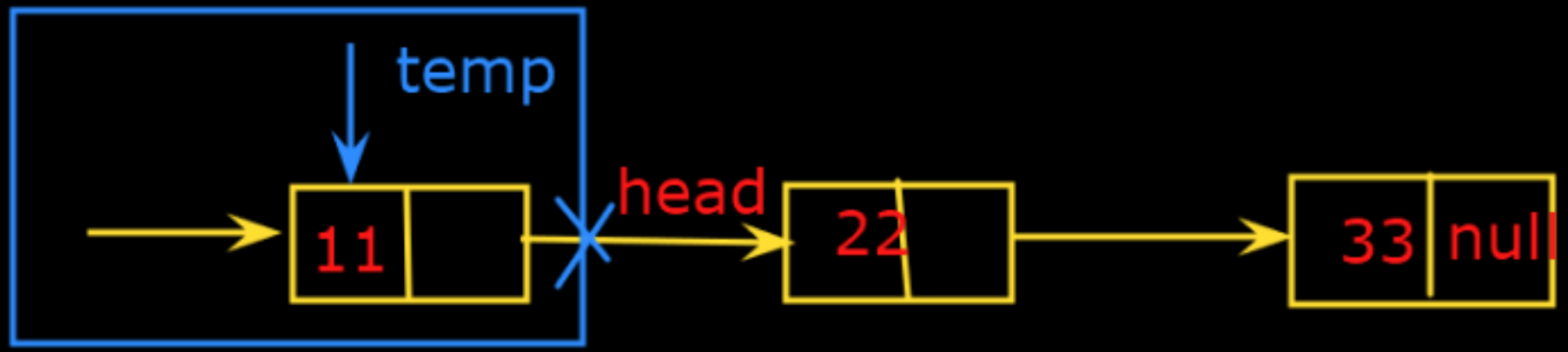
```
new_Node.next =n.next;
n.next = new_Node;
```



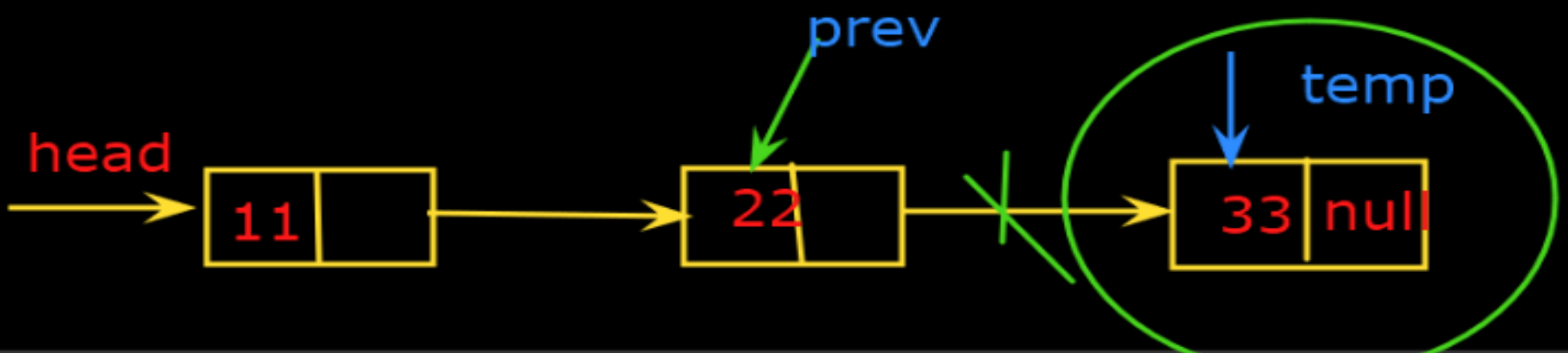
```
new_Node.next =n.next;
n.next = new_Node;
```




head
11
22
33
null




```
public void delete(int key)//11
{

Node temp = head;
if(temp != null && temp.data == key)
    head =temp.next;
    return;



}
```

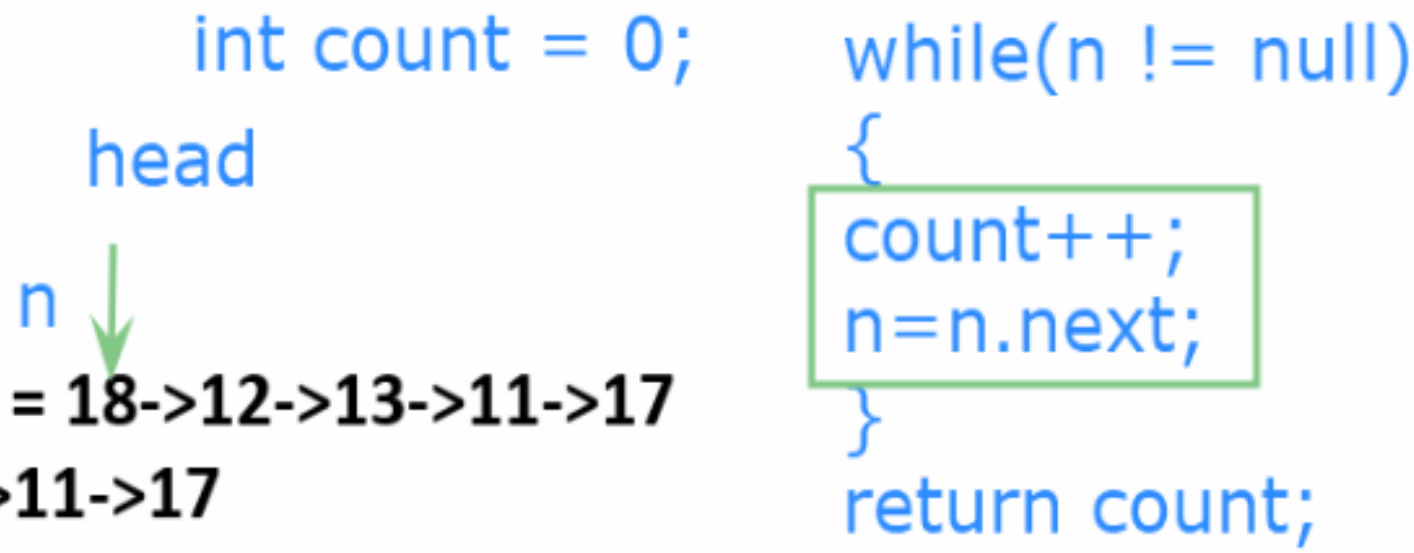



```
public void delete(int key)//33
{
Node temp = head, prev =null;
//case 1: Delete at first
if(temp != null && temp.data == key)
{
    head =temp.next;
    return;
}
//case 2 : Delete in between
while(temp != null && temp.data !==key)
{
```

Problem Statement 1 : Delete a Linked List node at a given position.

Given a singly linked list and a position, delete a linked list node at the given position.

Example:



Input: position = 1, Linked List = 18->12->13->11->17
Output: Linked List = 18->13->11->17

Input: position = 0, Linked List = 98->24->32->17->74
Output: Linked List = 24->32->17->74