Algorithms & Data Structure

Kiran Waghmare

Day 7: 10/11/2021

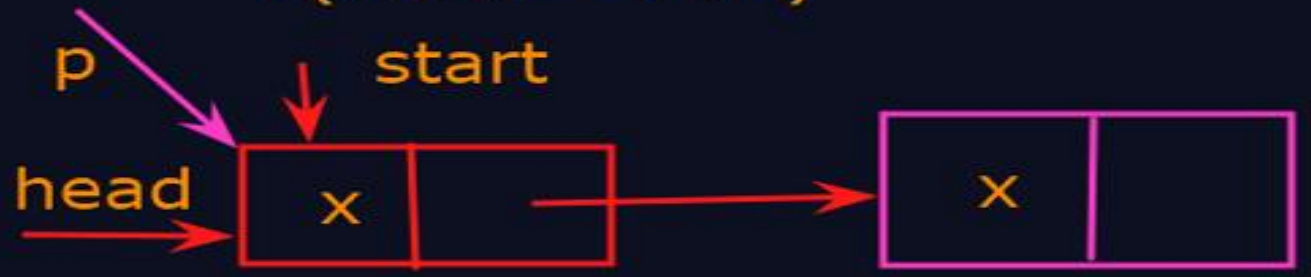
Topics:

-Linked List

-Linked List Operations

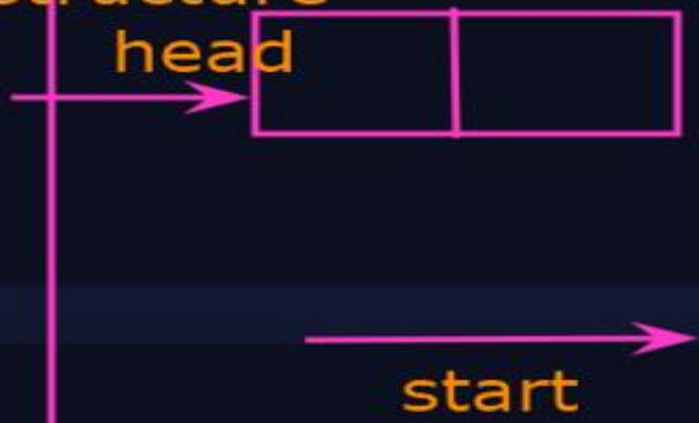
-Delete

```
int x=start.data;

int x = p.next.data;

head =start;

if(head==null)
```

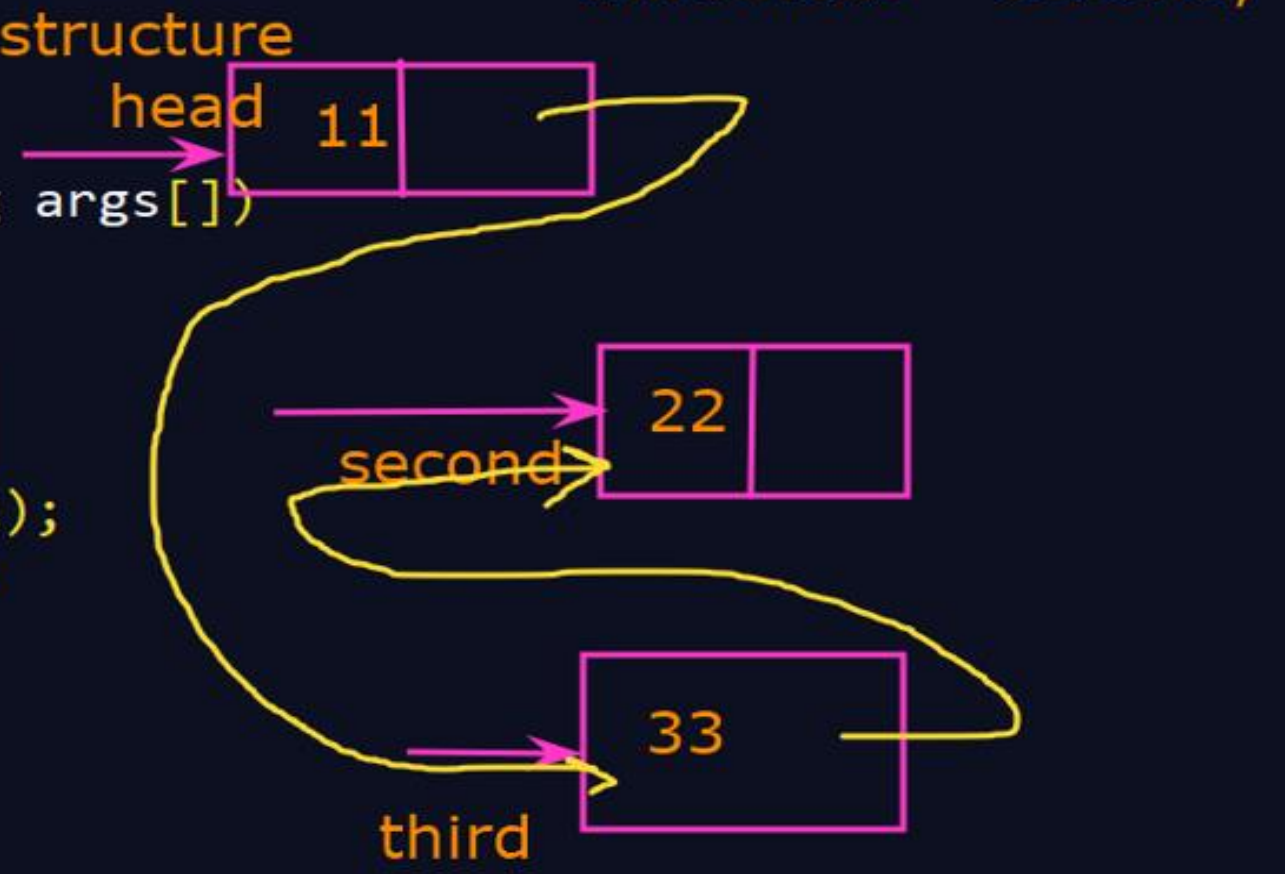


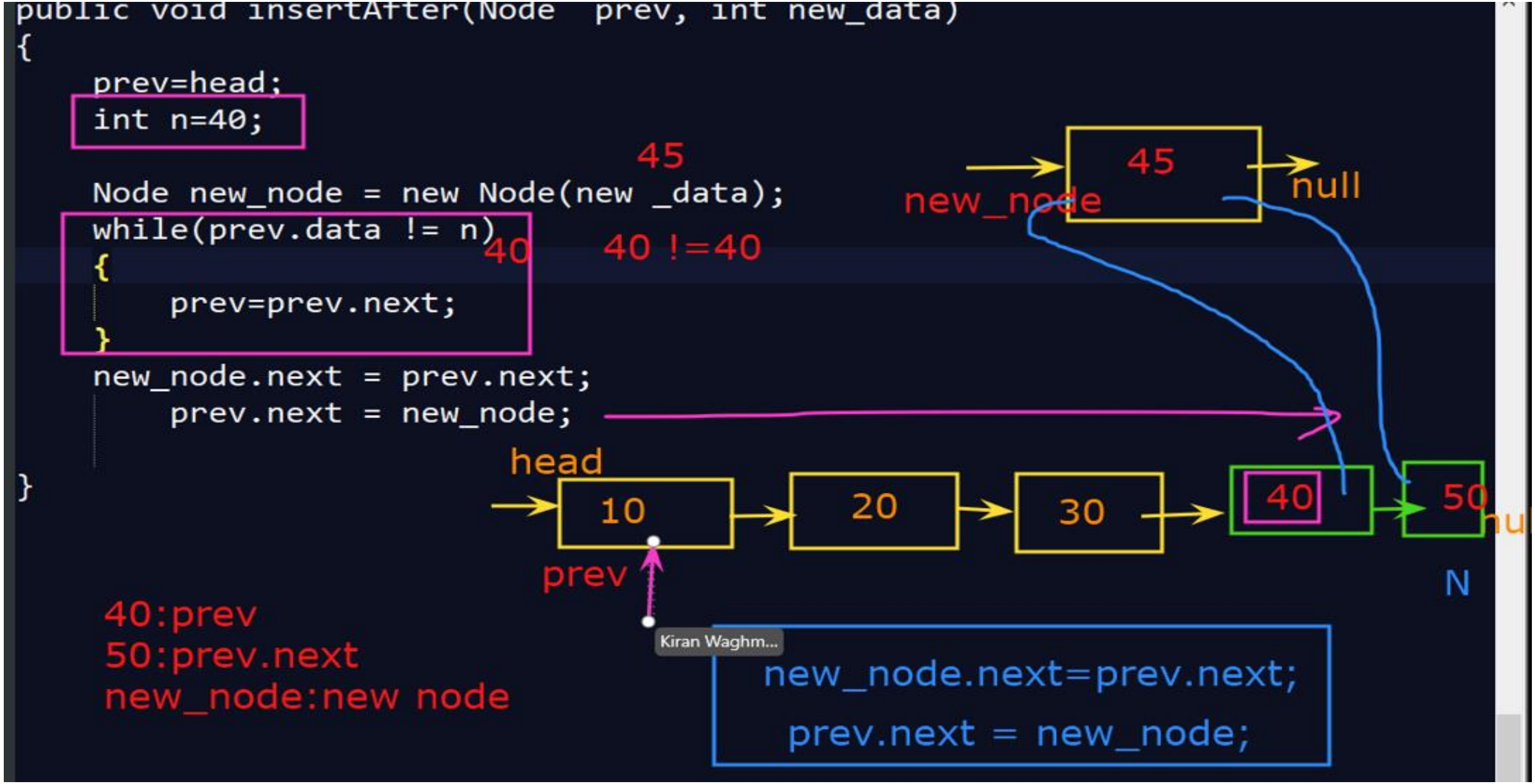

```
start = start.next

head.data

head.next.next

start=p;
```

```
public class List4 {
    Node head;//Start of list

    static class Node
    {
        int data;
        Node next;

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

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

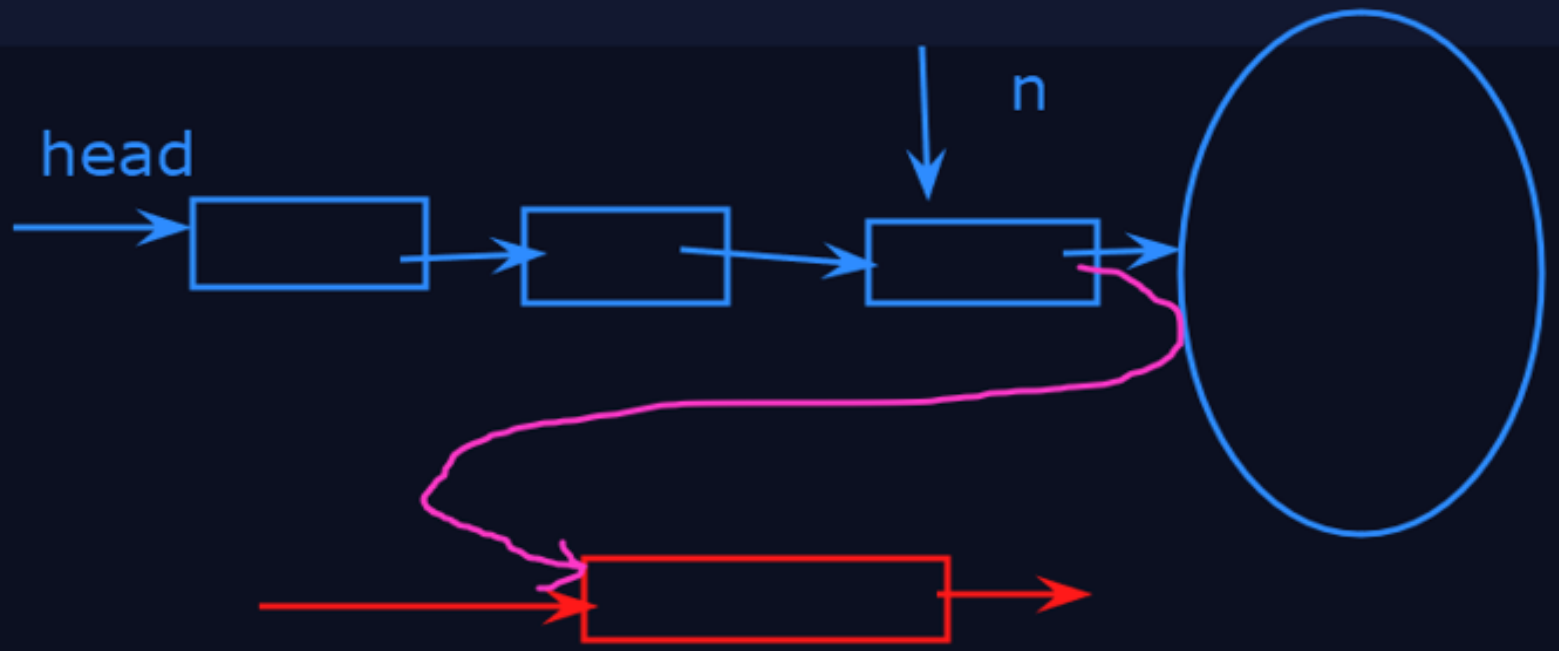


```
            last.next = new_node;
            return;


    }

public static void main(String args[])
{
    List4 l1 = new List4();

    l1.head = new Node(11);
    Node second  = new Node(22);
    Node third = new Node(33);

    l1.head.next = second;
    second.next = third;

    l1.display();// 11 22 33
    System.out.println(".....");
    l1.insert(44);// 44 11 22 33
    l1.insertAfter(l1.head.next, 55);//44 11 55 22 33
```




public void insertAfter(Node prev, int new_data)
{
prev=head;
int n=40;
Node new_node = new Node(new _data);
while(prev.data != n)
{
prev=prev.next;
}
new_node.next = prev.next;
prev.next = new_node;
}
45
40
40 !=40
45
new_node
null
head
10
20
30
40
50
prev
N
40:prev
50:prev.next
new_node:new node
Kiran Waghm...
new_node.next=prev.next;
prev.next = new_node;

```
    if(head == null)
    {
        head = new Node(new_data);
        //head = new_node;
        return;
    }

    Node n = head;
    while(n.next != null)
        n=n.next;
    n.next=new_node;
    return;
}
```

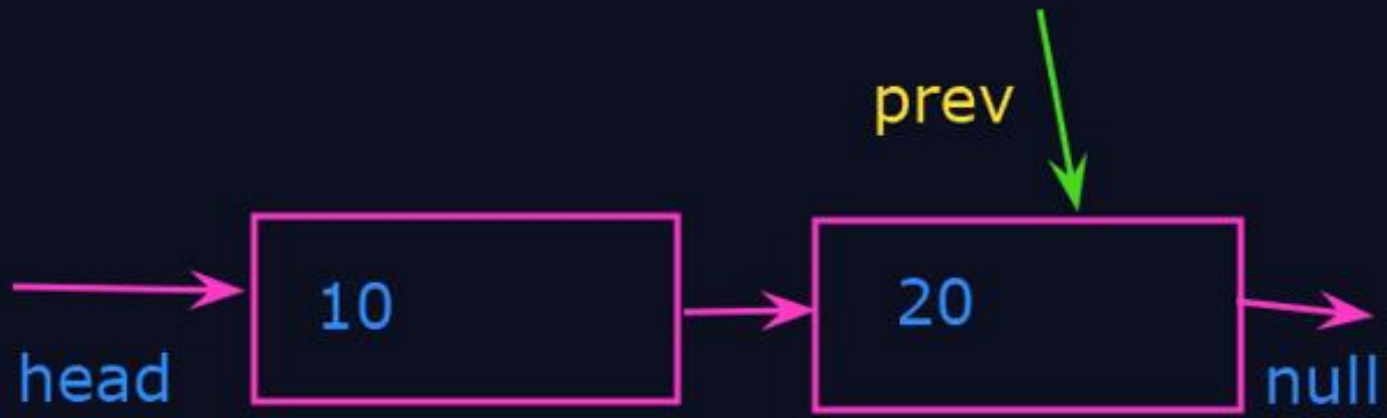


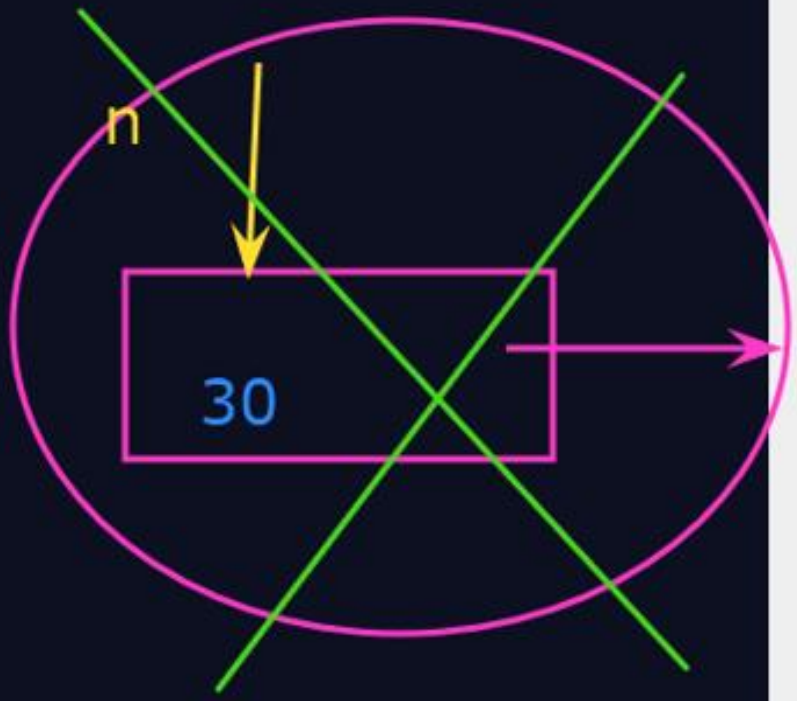
Case 3: Insertion of between 2 nodes

n.next = new_node;

-Linked List Operations

-Delete

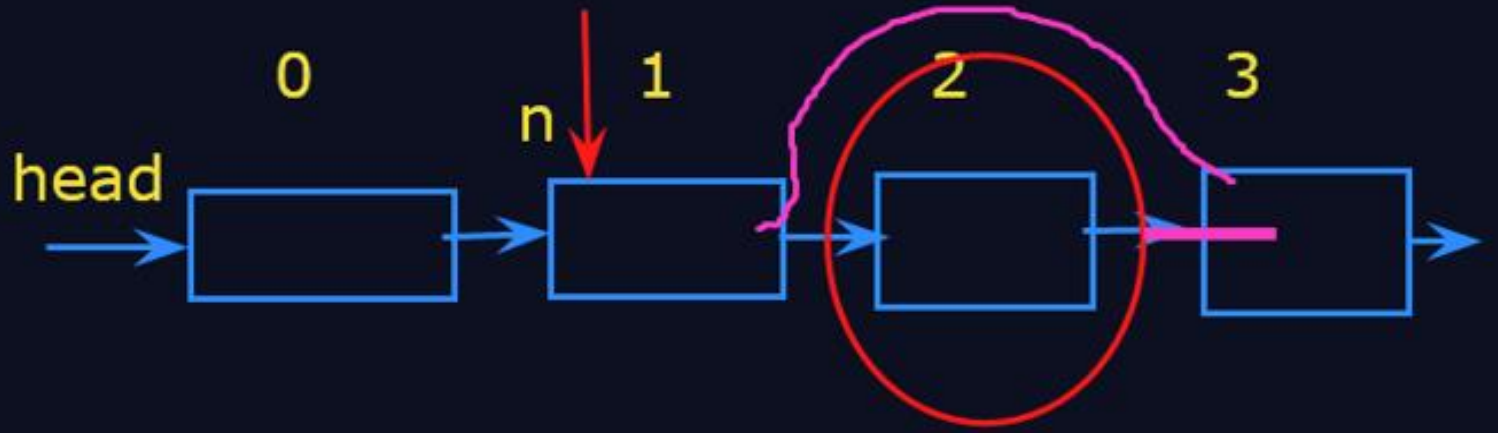




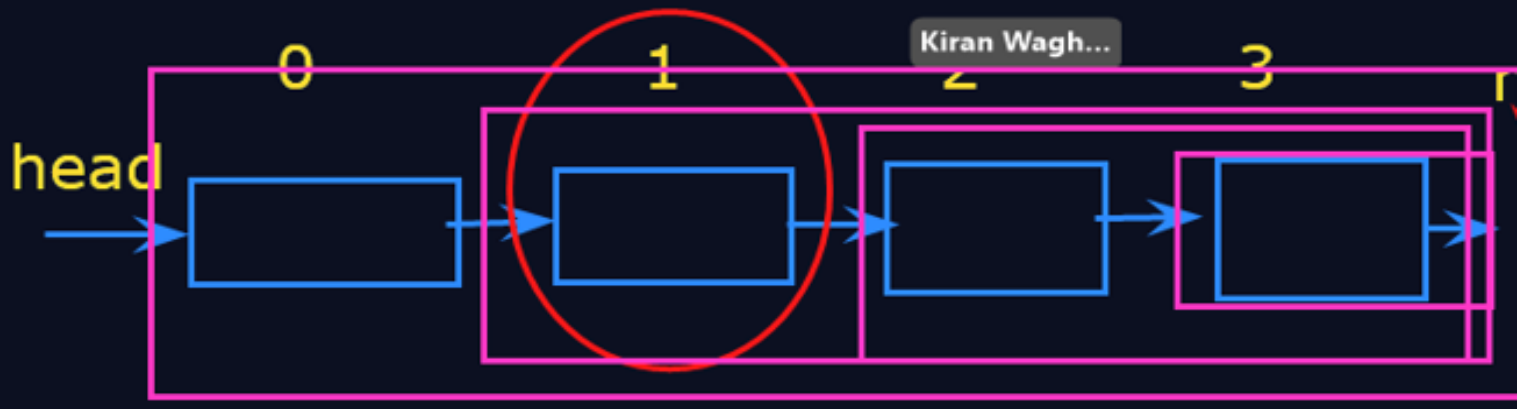
prev.next=null;

prev.next = n.next;

```
{
    if(head == null)
        return;
    Node n = head;
    if(pos == 0)
    {
        head = n.next;
    }

    for(int i=0;n != null && i < pos-1;i++)
        n=n.next;
    if(n == null)
        return;
        n.next = n.next.next;


}
```



```
    int c = 0;
    while(n != null)
    {
        c++;//1, 2, 3, 4
        n = n.next;
    }
    return c;
}
```



0
1+0=1
1+1=2
1+2=3
1+3=4

Using Recursion:

```
int count(Node n)
{
    if(n == null)
        return 0;

    return 1+ count()

}
```

```
        return 0;

    return 1+ count(n.next);

}

----------------------------
Search in Linked List:

boolean search(Node head, int key)
{
    Node n = head;
    while(n != null)
    {
        if(n.data  == key)
            return true;
        n=n.next;
    }
    return false;

}
```

0
1+0=1
1+1=2
1+2=3
1+3=4

Thanks