Sep22 : Day 6

Kiran Waghmare

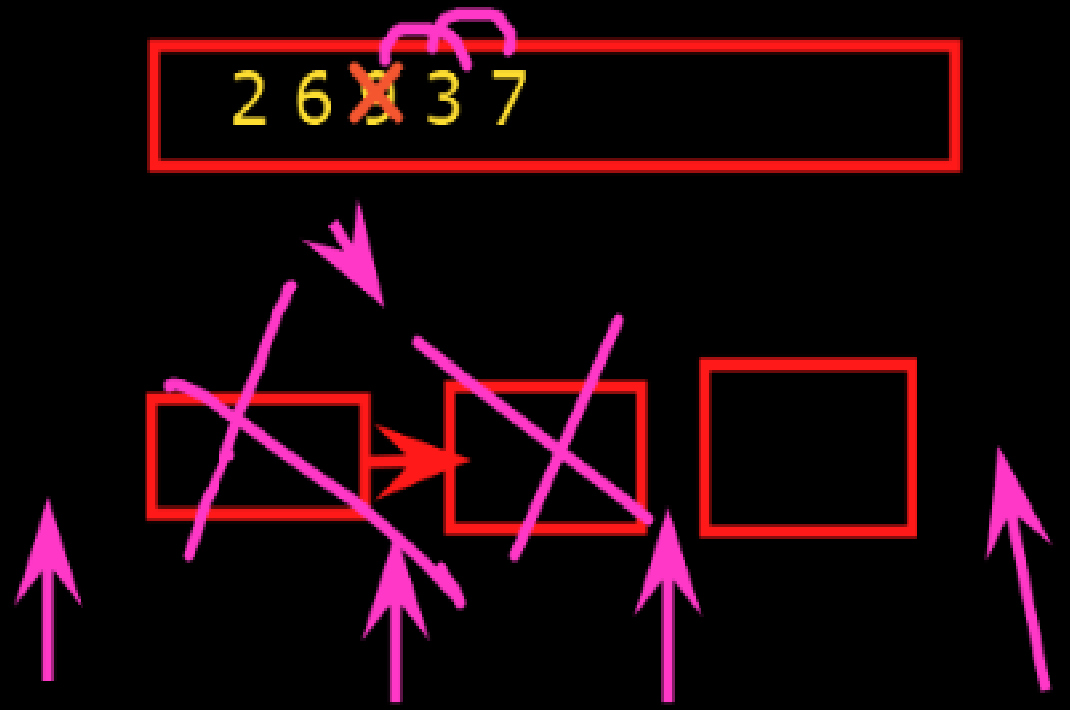
CDAC Mumbai

}

Linked list:

-sequence of data structures, which are connected together via links.

- -sequence of links
- -connections between nodes
- -most used data structure
- -provides lot of flexibility



Terms:

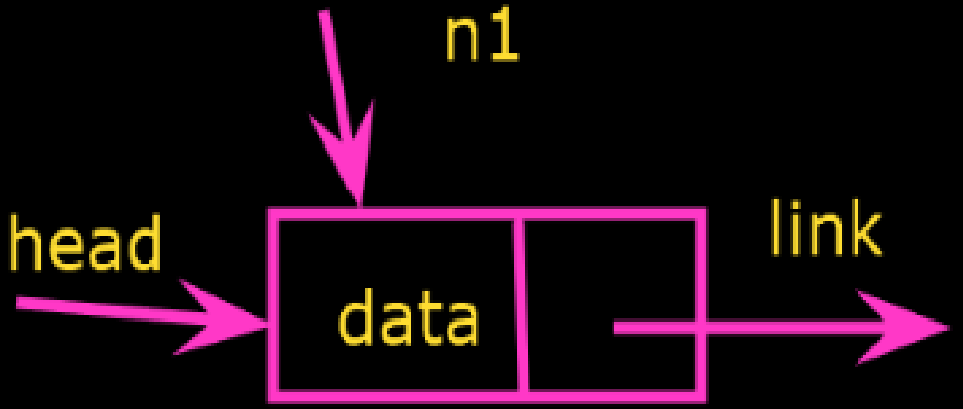
- -Link :data=element, link=address
- -Next : next is a link: address
- -Data : any primitive data types
- -Linked list : connection of links :

- -First node of linked list = starting node of list
- -Last node of linked list = link is null
- -chain of nodes....

Linked list:

-sequence of data structures, which are connected together via links.

- -sequence of links
- -connections between nodes
- -most used data structure
- -provides lot of flexibility



Terms:

- -Link :data=element, link=address
- -Next : next is a link: address
- -Data : any primitive data types
- -Linked list : connection of links :

head=~~3000~~

head=n1

- -First node of linked list = starting node of list
- -Last node of linked list = link is null
- -chain of nodes....

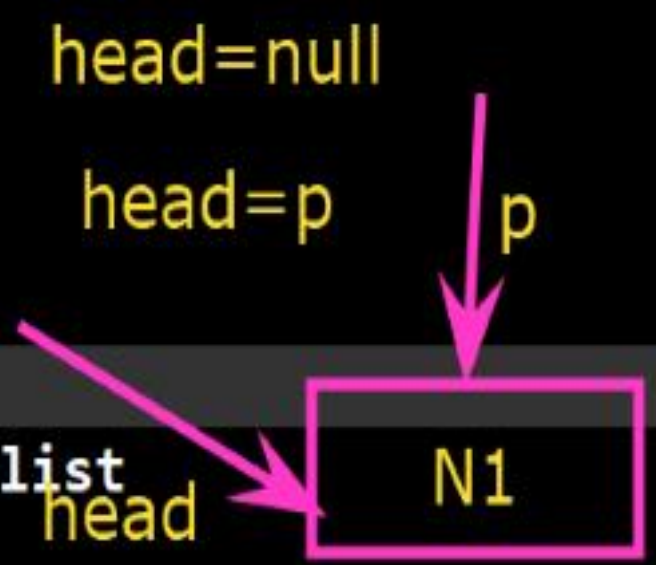
Linked list:

-sequence of data structures, which are connected together via links.

- -sequence of links
- -connections between nodes
- -most used data structure
- -provides lot of flexibility

Terms:

- -Link :data=element, link=address
- -Next : next is a link: address
- -Data : any primitive data types
- -Linked list : connection of links :
- -First node of linked list = starting node of list
- -Last node of linked list = link is null
- -chain of nodes....



Who can see what you share here? Recording On

```java
class List1
{
    Node head;

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

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

    public static void main(String args[])
    {

        List1 l1 = new List1();
        l1.head = new Node(11);
        Node second = new Node(22);
        Node third = new Node(33);

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

        l1.display();
    }
```

Node structure banan hai!

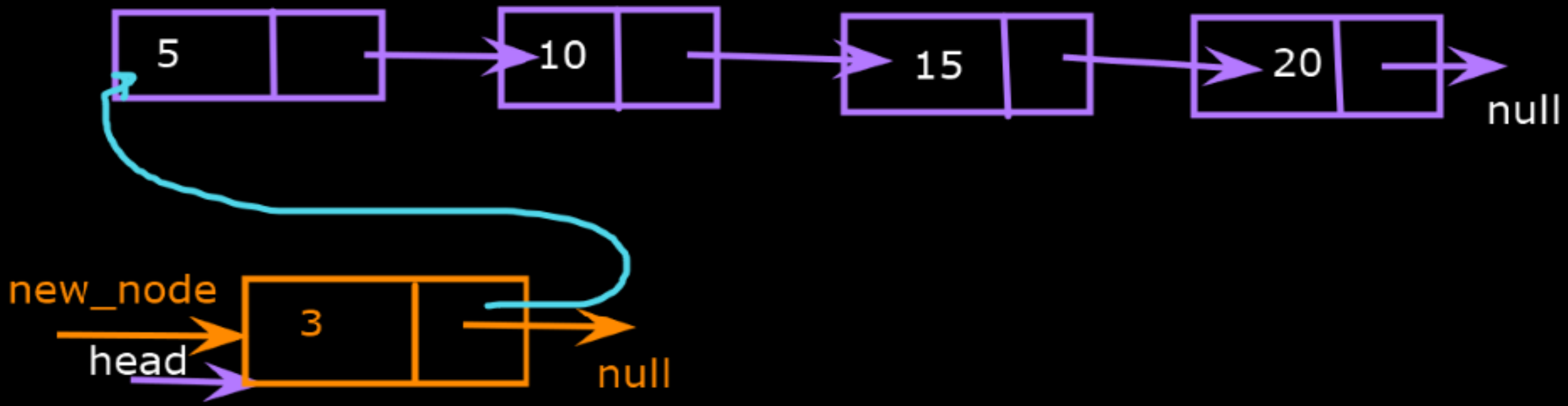
Default values dena tha!

Time : Best, Worst, Average

O(1)

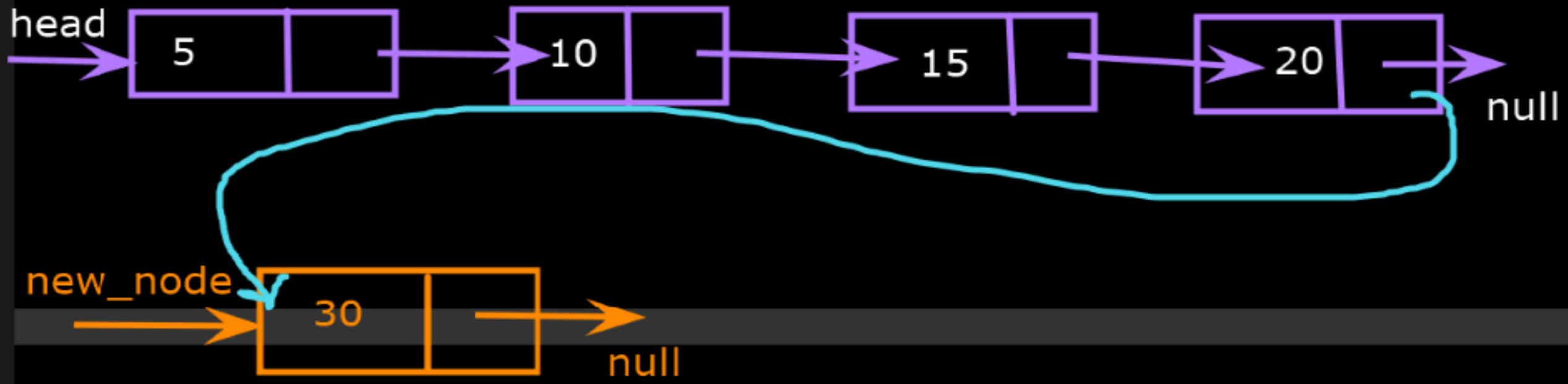
Auxillary space: O(n)

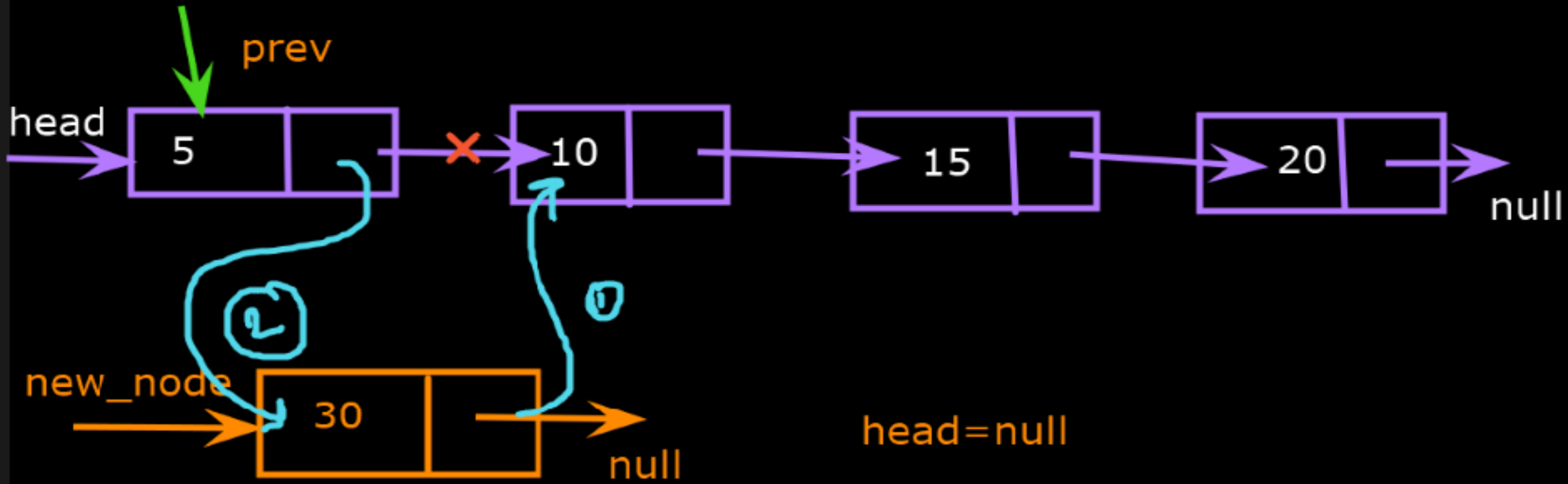
ccase 1:




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


}
```

case 2: at the end



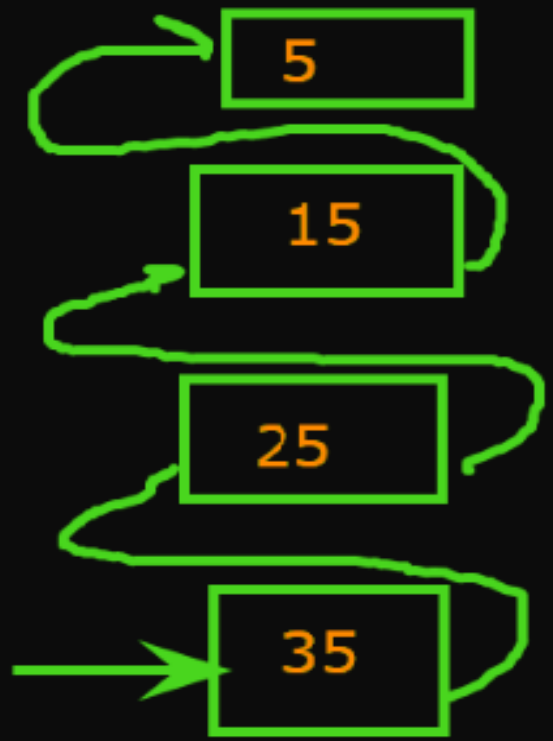



```
static void insertAfter(Node prev, int new_data)
{
    Node new_node = new Node(new-data);
    //Follow the sequence
        new_node.next = prev.next;//1
        prev.next = new_node;//2

}
```

```
            n=n.next;
        }
    }

    public static void main(String
    {

        List2 l1 = new List2();

        l1.display();
        System.out.println("Empty L
        insert(5);
        insert(15);
        insert(25);
        insert(35);
        l1.display();
    }

}
```

C:\WINDOWS\system32\cmd.exe

```
C:\Test>java List2
Empty List !!!
5--->
C:\Test>javac List2.java

C:\Test>java List2
Empty List !!!
35--->25--->15--->5--->
C:\Test>
```



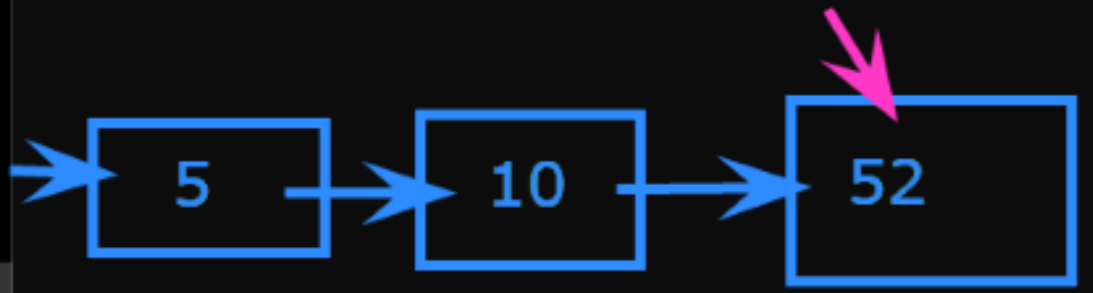
```
{
    Node temp = head, prev =null;
    //case 1 : delete at begining
    if(temp !=null && temp.data == key)
    {
        head=temp.next;//head shift karega
        return;
    }
    //remaining cases
    while(temp != null && temp.data != key)
    {
        prev =temp;
        temp=temp.next;

    }
    if(temp == null)
        return;
    prev.next = temp.next;


}

    void display()
    {
        Node n = head;
        while(n != null)
```

Mouse Select Text Draw Stamp Spotlight Eraser Format Undo Redo Clear

Who can see what you share here? Recording On

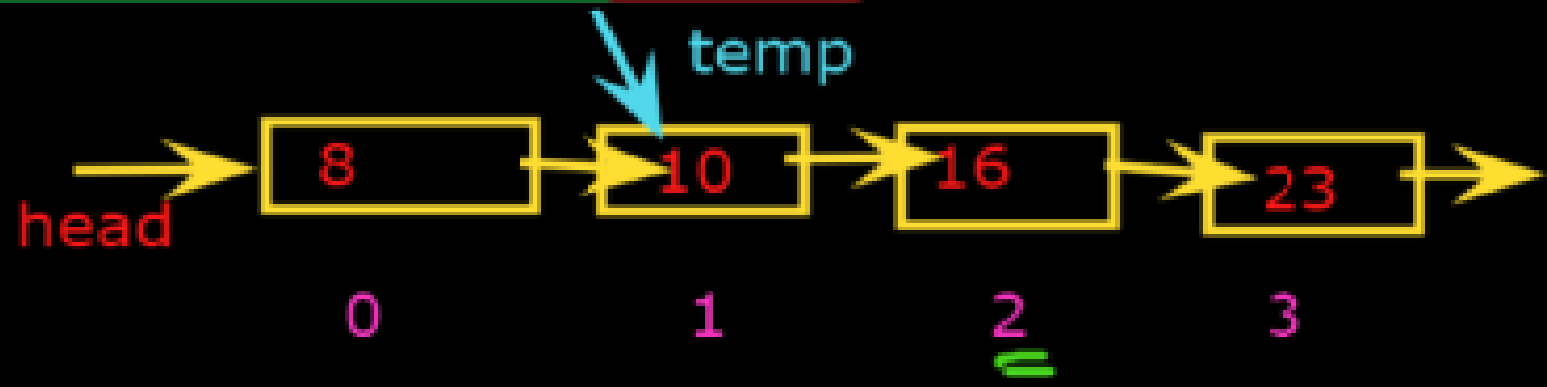
C:\Test>

key =1

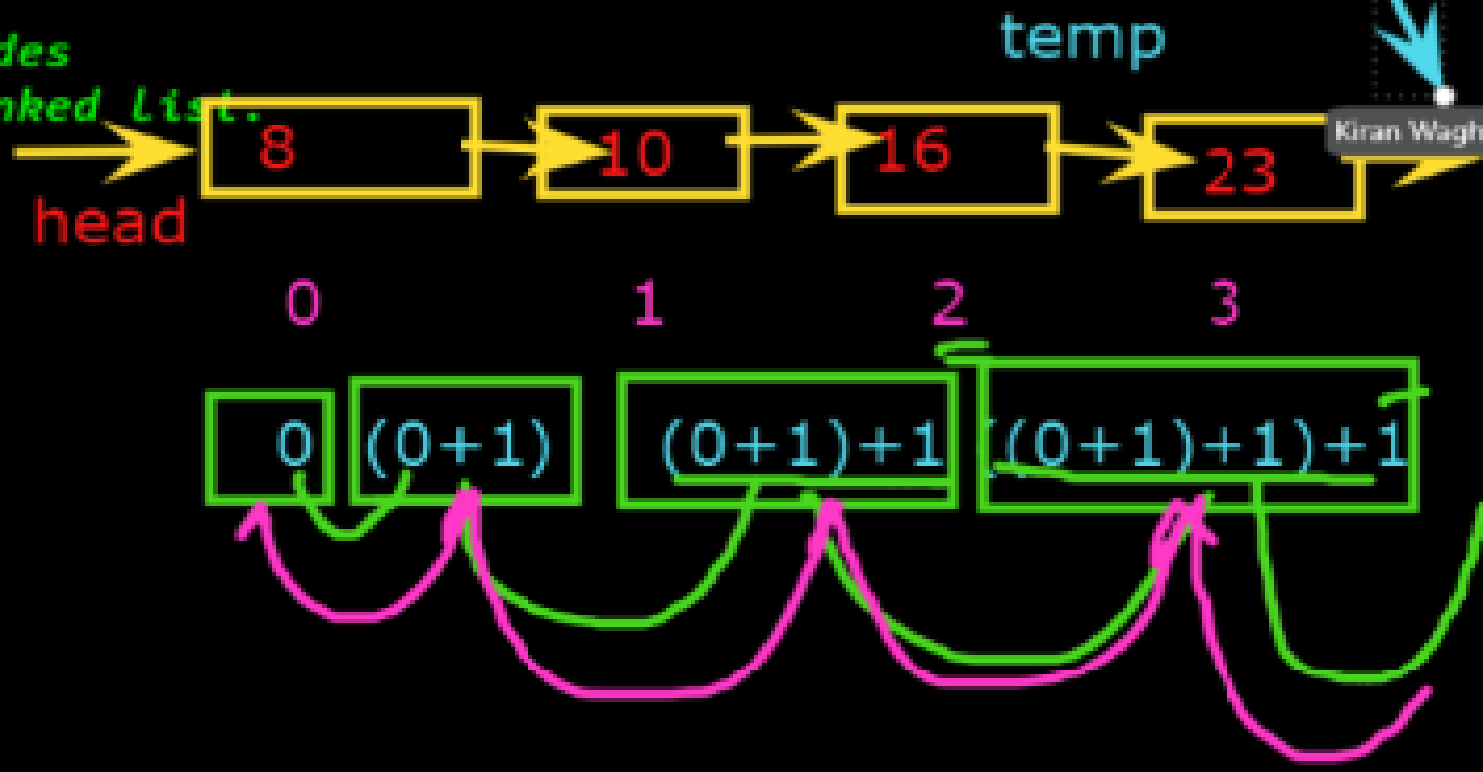


Kiran Wagh...

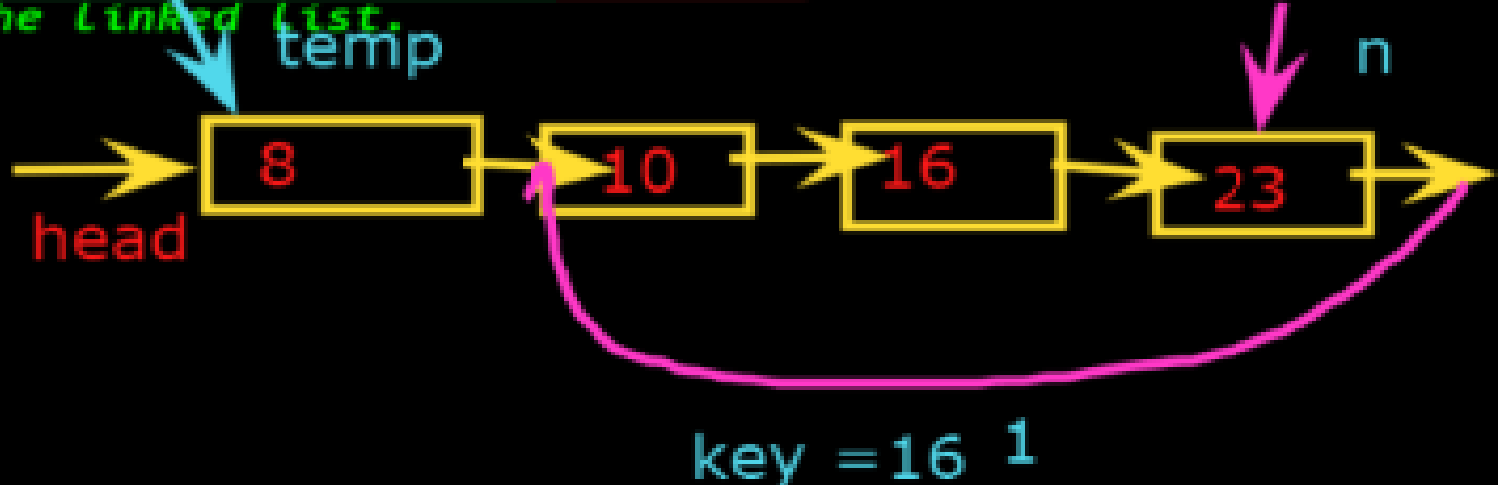
```java
//Deletion at particular position.

static void deleteNode(int pos)
{
    //List is empty ?
    if(head == null)
        return;
    Node temp = head;
    if(pos == 0)
    {
        head=temp.next;
        return;
    }


    for(int i=0;temp != null && i<pos-1;i++)
    {
            temp = temp.next;
    }
    if(temp == null || temp.next == null)
        return;
    Node n = temp.next.next;
    temp.next = n;
}
```



```java
//function to count the number of nodes
//function to count the length of linked list.
static int count()
{
    Node temp = head;
    int c=0;
    while(temp !=null)
    {
        c++;
        temp=temp.next;
    }
    return c;

}

//recursive
int countrecursive(Node temp)
{
    //base condition
    if(temp = null)
        return 0;

    return 1+countrecursive(temp.next)
}
```



```java
//function to search an element in the linked list.

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

    }
    return false;
}
```



Home work:

1. function to identify the Reverse a linklist
2. function to identify the nth node from the linked list.
3. function to identify the nth node from end of the linked list.
4. function to identify the middle of the linked list.
5. function to identify the loop in the linked list.