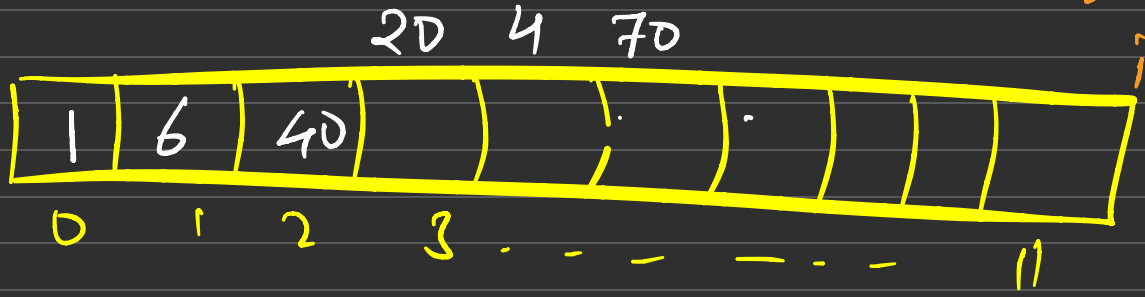
Doing the best at this moment, puts you in the best place for the next moment.

If we have Array, then Why Linked List

x = 20 int 3

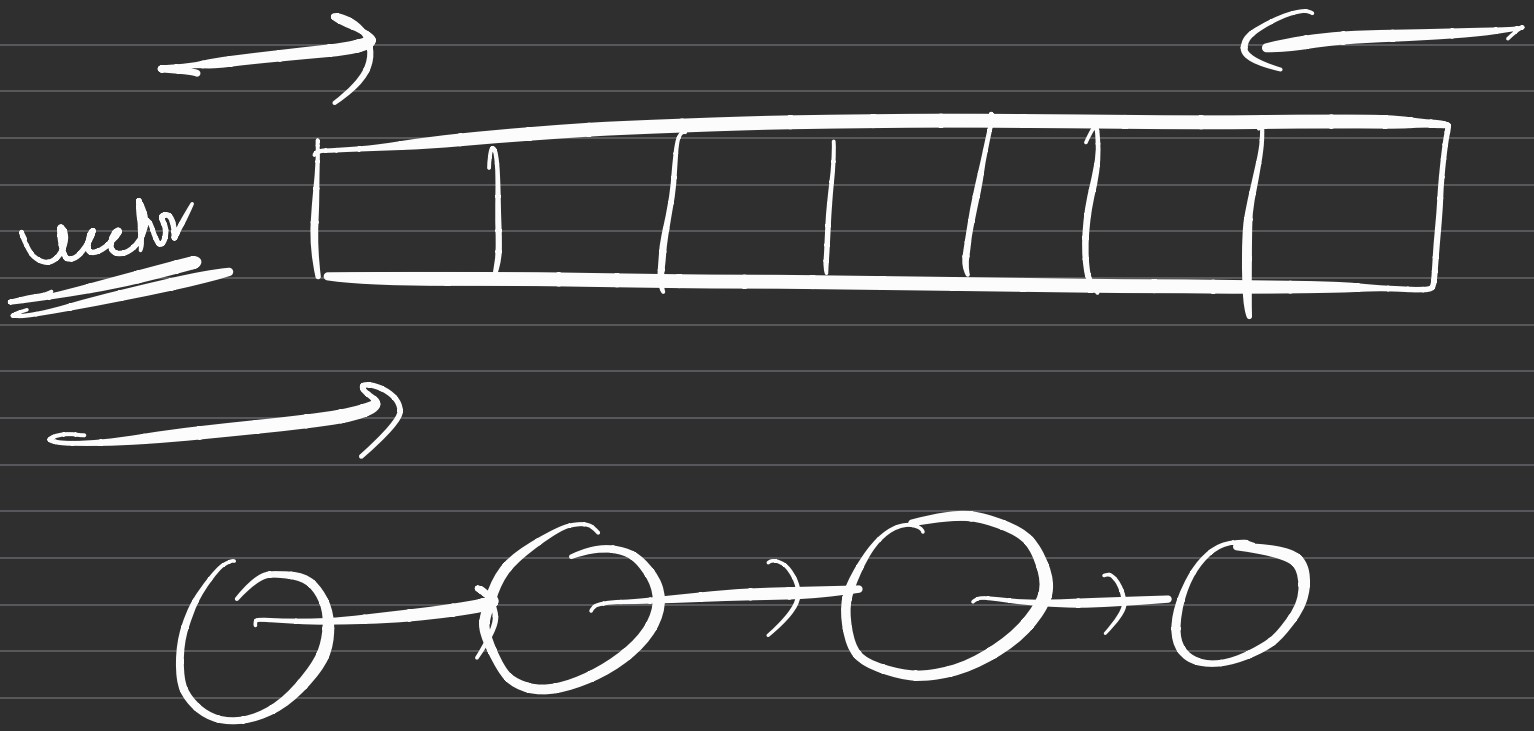
insertion / deletion → difficult

x = 70

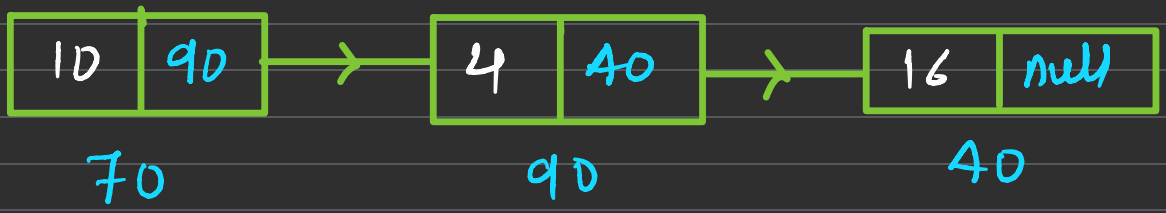


x = 40 at index 2

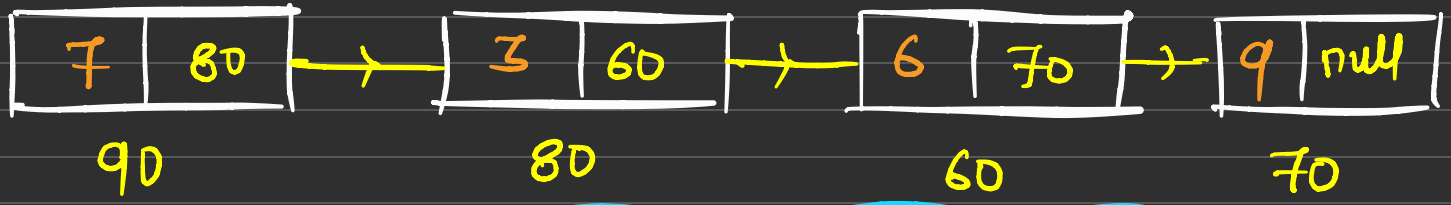
N = 12



What is Linked List



Collection of data which is represented in terms of Node



Pointer → Address of Next Node

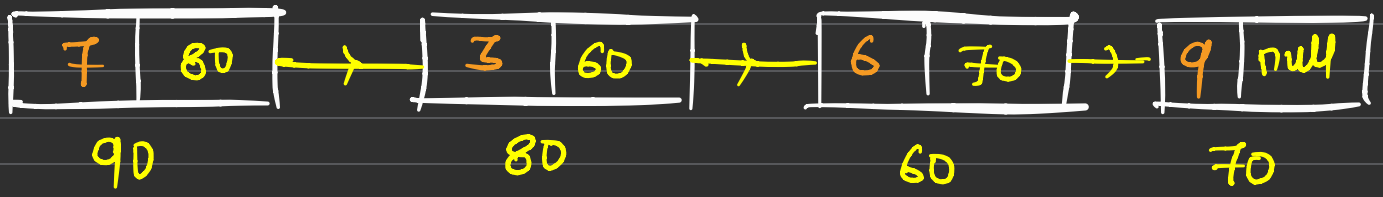
Data

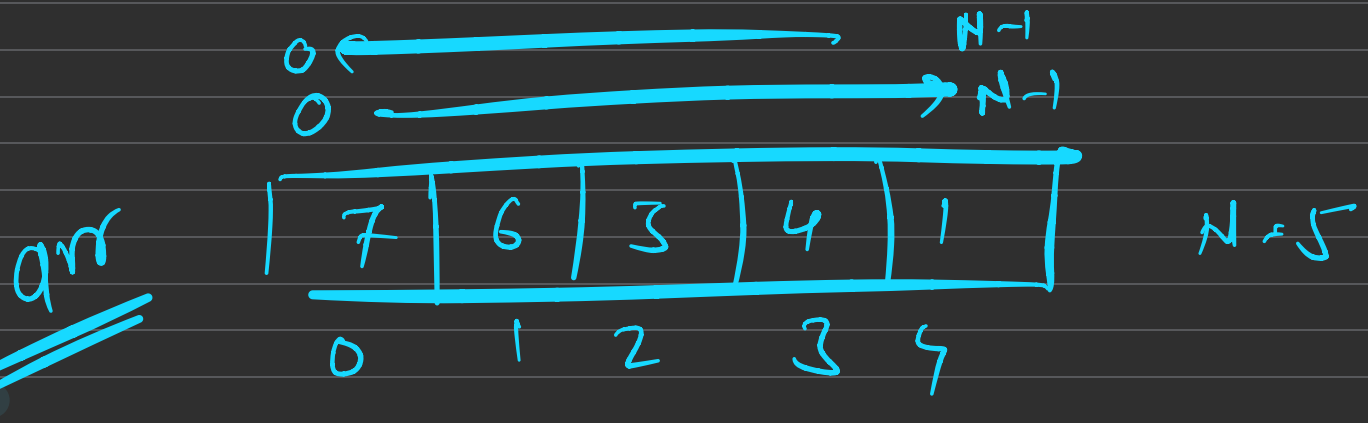
- string
- int
- char
- float
- double

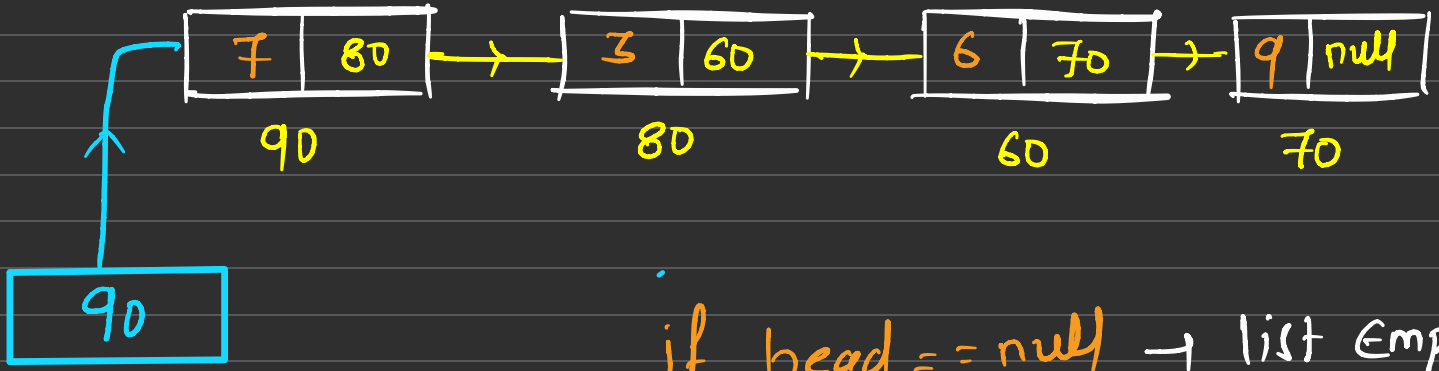
Address

- Address of next Node
- Null → if no next Node

Head / Start Node







if head == null → list Empty

Not to change position of head

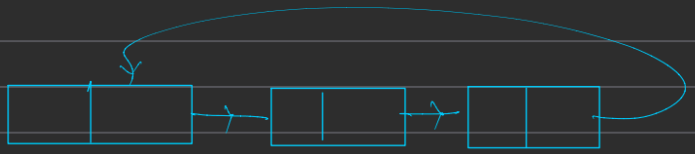
head ⇒ start of LL

Types of Linked List

Singly Linked List

Doubly Linked List

Circular Linked List



- Types of operation:

* Insertion
  - start
  - end
  - Position
* Deletion
  - start
  - end
  - Position
* Traversal

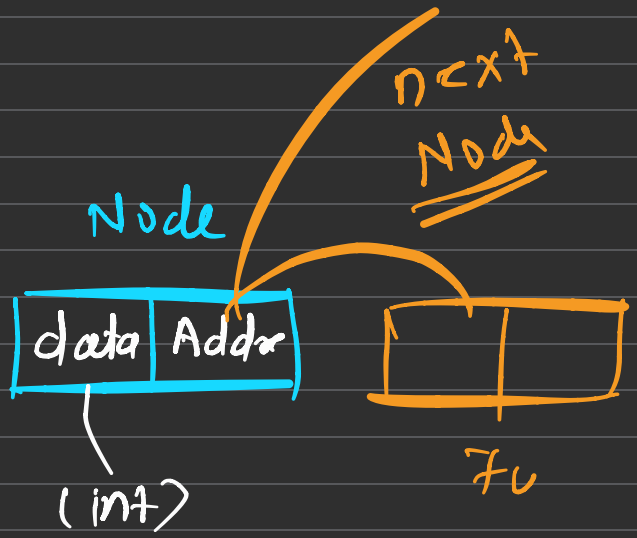
Structure of SLL

C++

```
Class Node
{
    int data
    Node *next
};
```

Java

```
Class Node
{
    int data
    Node next
};
```



How to create Node

* new keyword → to allocate memory during RT

CPP = Node *node = new Node()

Java = Node node = new Node()

CPP

```
class Node
{
    int data
    Node * next

    Node (int data)
    {
        this -> data = data
        next = null
    }
};
```

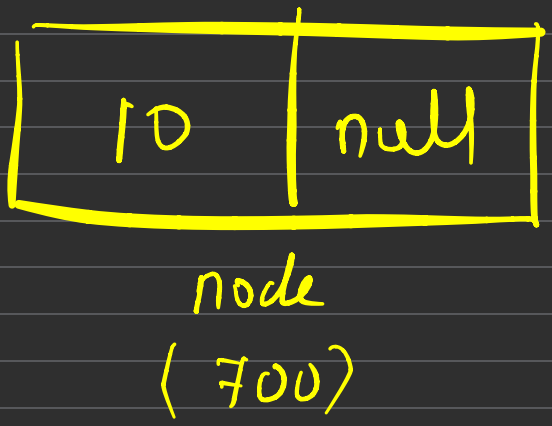
Java

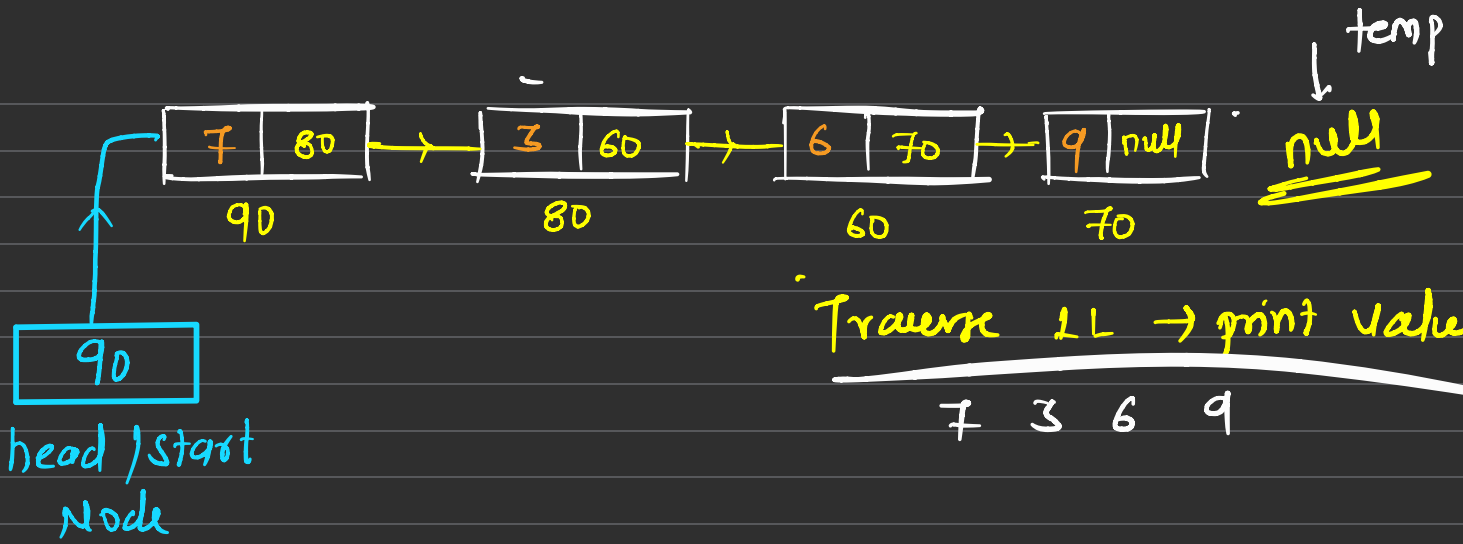
```
class Node
{
    int data
    Node next

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

CPP = Node *node = new Node(10)

Java = Node node = new Node(10)





Traverse LL → print value

7 3 6 9

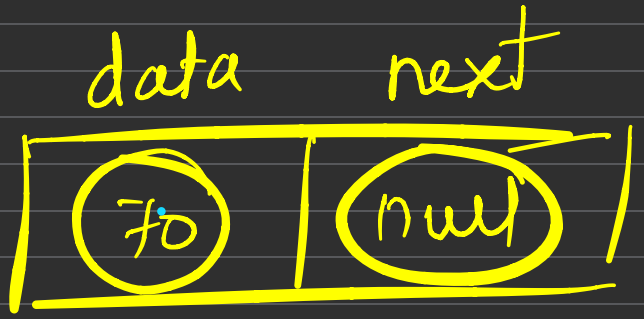
```
Node * temp = head
while ( temp != null )
{
    print ( temp → data)
    temp = temp → next
}
```

CPP

```
Node temp = head
while ( temp != null )
{
    print ( temp.data)
    temp = temp.next
}
```

Java

node → next
node → data



node

(&node).next

(*node).data

node.next ✓

node.data ✓

ताली बजाने से पहले तानों का
दौर आता है,
ये रिवाज हर कलाकार ने
सिखाया है!!

Insertion (start)

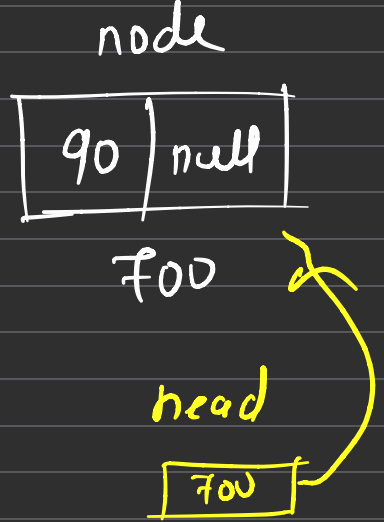
Node * node = new Node(70)

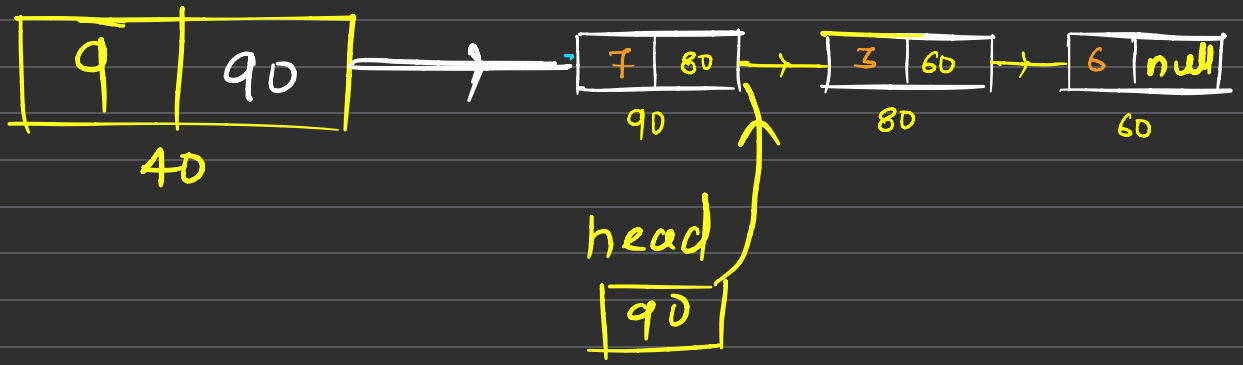
node → next = head

head = node

```
Node * node = new Node(70)

if (head == null)
{
    head = node
}
else
{
    node->next = head
    head = node
}
```





Class Node

{

data

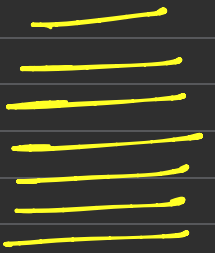
next

}

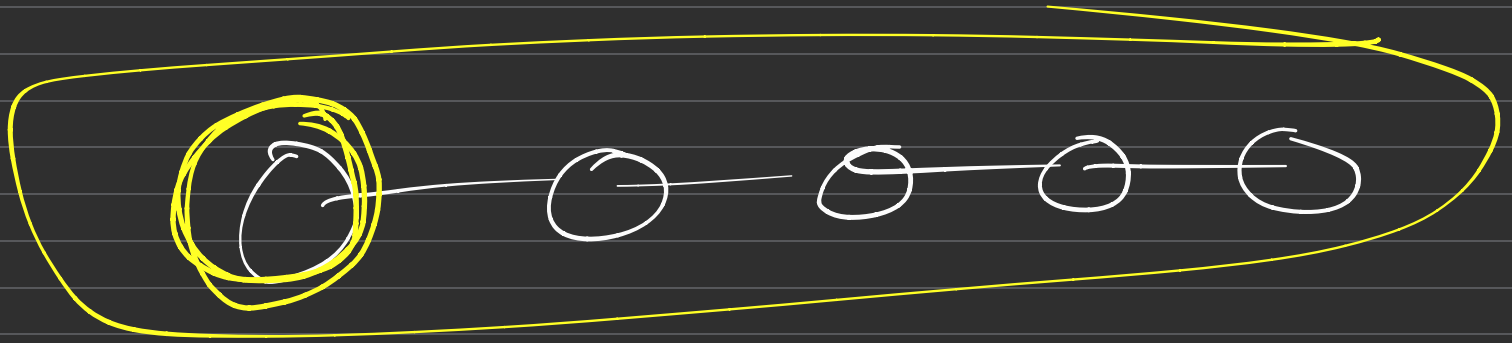
node

Class SLL

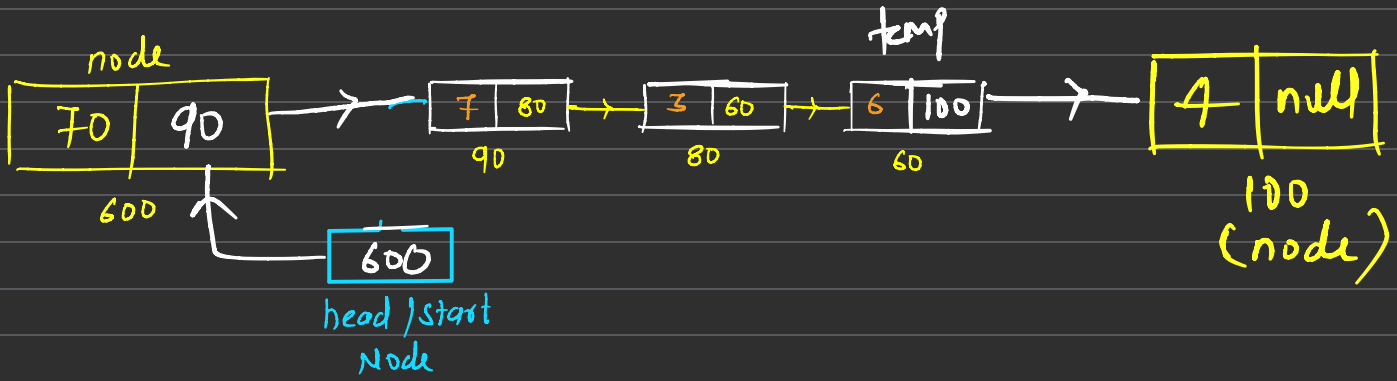
{



}



Insert (End)

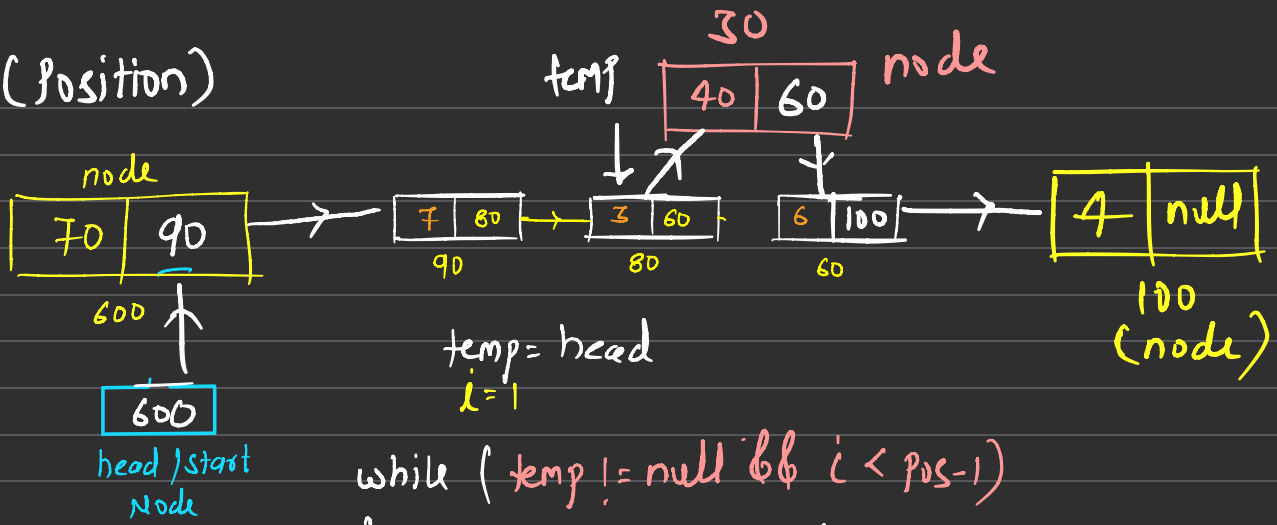



```
Node * node = new Node(4)
```

temp

```
if(head==null)
{
  head=node
}
else
{
  temp=head
  while(temp->next!=null)
  {
    temp=temp->next
  }
  temp->next=node
}
```

Insert (Position)



temp = head
i = 1

```
while ( temp != null && i < pos-1)
{
    temp = temp -> next
    i++
}
```

node -> next = temp -> next

temp -> next = node

```
if(pos==1)
{
    start()
}
else if(pos==cnt+1)
{
    end()
}
```

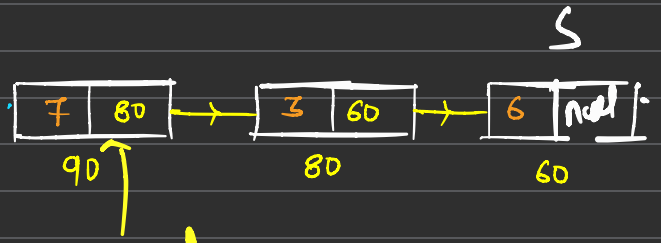
की जिंदगी में एक समय आता है
जब उसे तय करना होता है
की पंन्ना पलटना है
या किताब बंद करनी है!!

Deletion (start)

Bucket

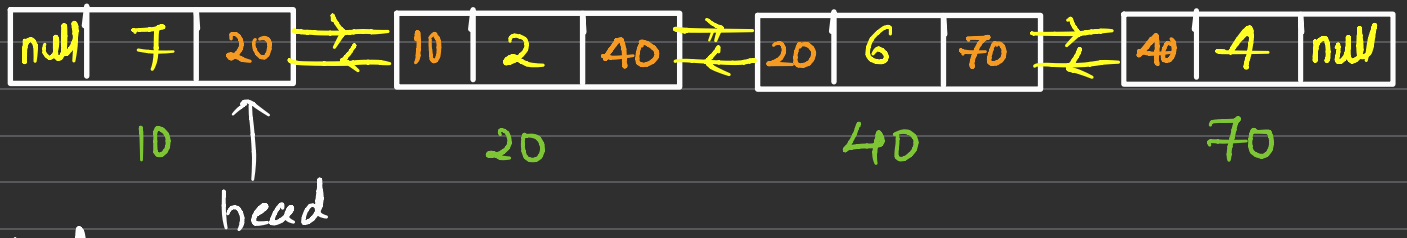
```
if (head == null)
    return
temp = head
head = head.next
delete (temp)
```

Delete (End)



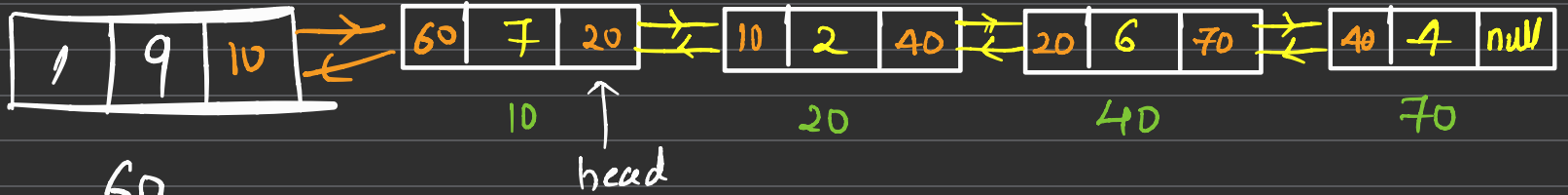

```
fast = head
slow = head
while ( fast->next != null )
{
    slow = fast
    fast = fast->next
}
slow->next = null
delete ( f )
```

DLL




```
Class Node
{   int data
    Node * next
    Node * prev
    Node (int data)
    {   this → data = data
        next = prev = null
    }
}
```

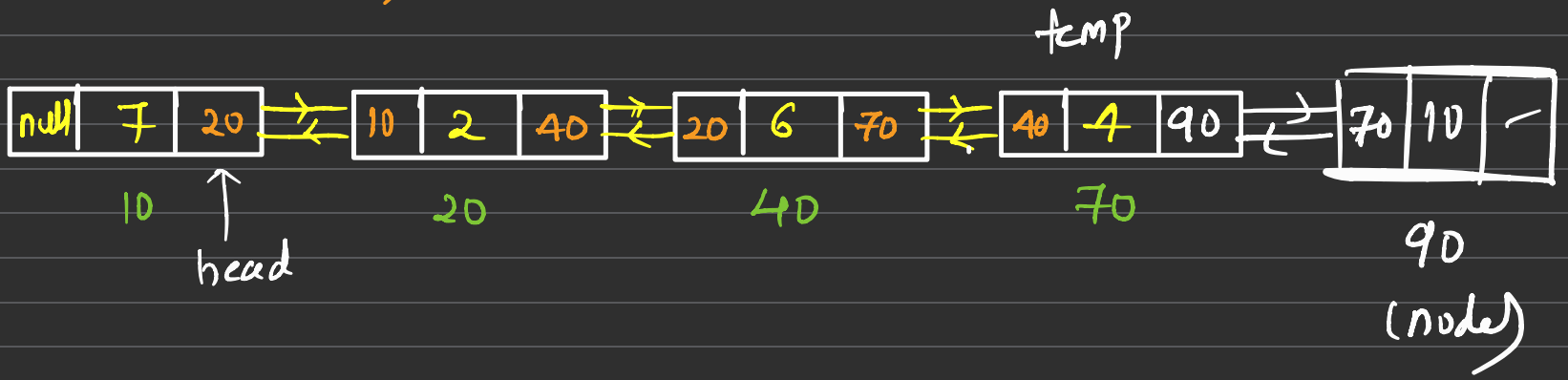
Insert (start)



60
(node)

```
if (head == null)
    head = node
node → next = head
head → prev = node
```

Insert (End)



temp → next = node

node → prev = temp

- Insert by position → DLL
- delete by position → SLL
- Traverse from end → DLL

Homework