

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



Introduction to Linked List



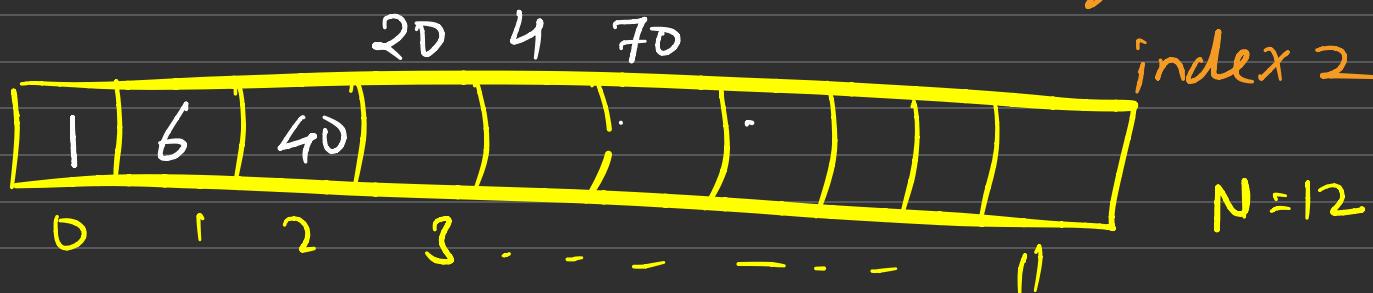
If we have Array, then Why Linked List

~~x = 20~~ int 3

insertion / deletion → difficult

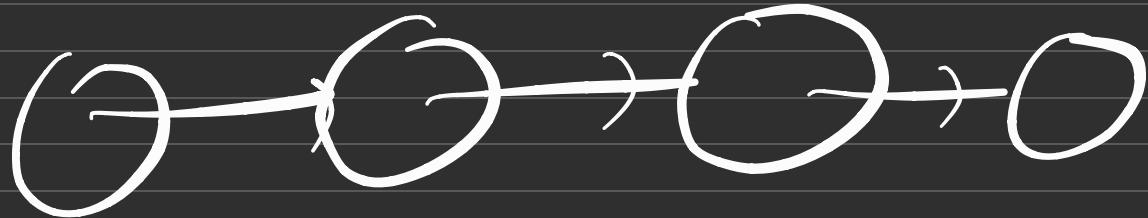
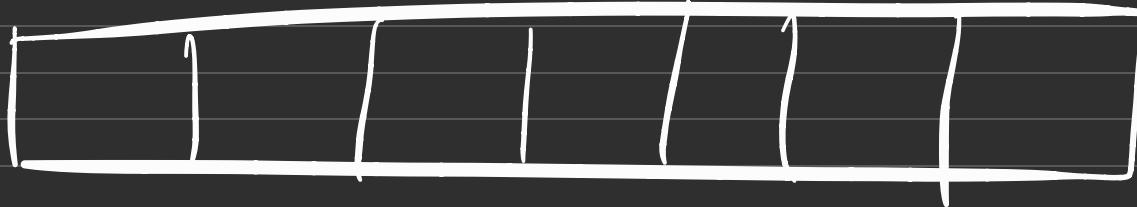
~~x = 70~~

x = 40 at
index 2





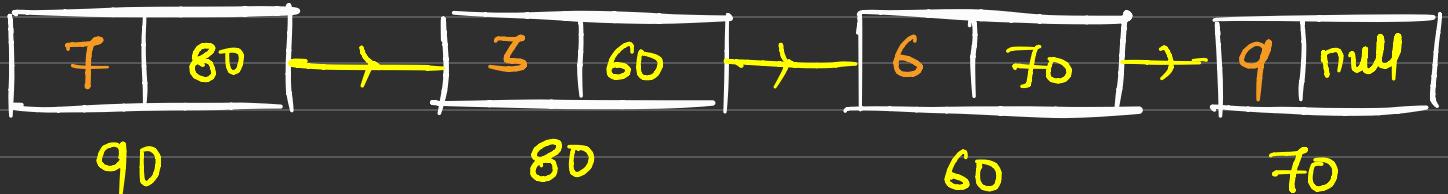
vector



What is Linked List



Collection of data which is represented in terms of Node



Pointer

Address of
next Node



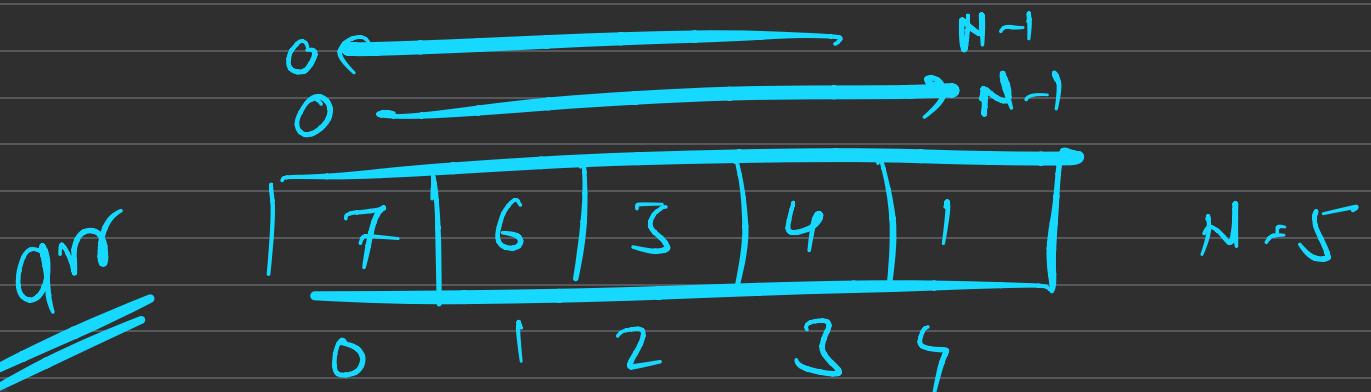
Data {

- string
- int
- char
- float
- double

Address {

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

Head | Start Node





if $\text{head} == \text{null}$ → list Empty

Not to change position of head

head \Rightarrow start of LL

Types of Linked List

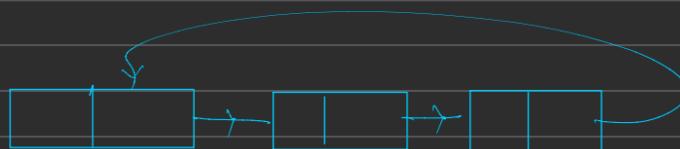
Singly Linked List



Doubly Linked List



Circular Linked List



- Types of operation:

* Insertion

- Start
- end
- Position

* Traversal

* Deletion

- start
- end
- Position

Structure of SLL

C++

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

Java

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

Node



next

Node



fu

How to create Node



* new keyword → to allocate memory during RT

CPP = Node *node = new Node()

Java = Node node = new Node()

CPP

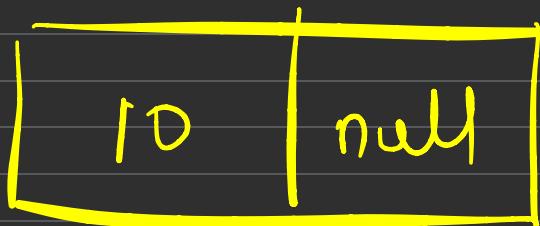
```
class Node  
{  
    int data  
    Node *next  
  
    Node (int data)  
    {  
        this->data = data  
        next = null  
    }  
}
```

Java

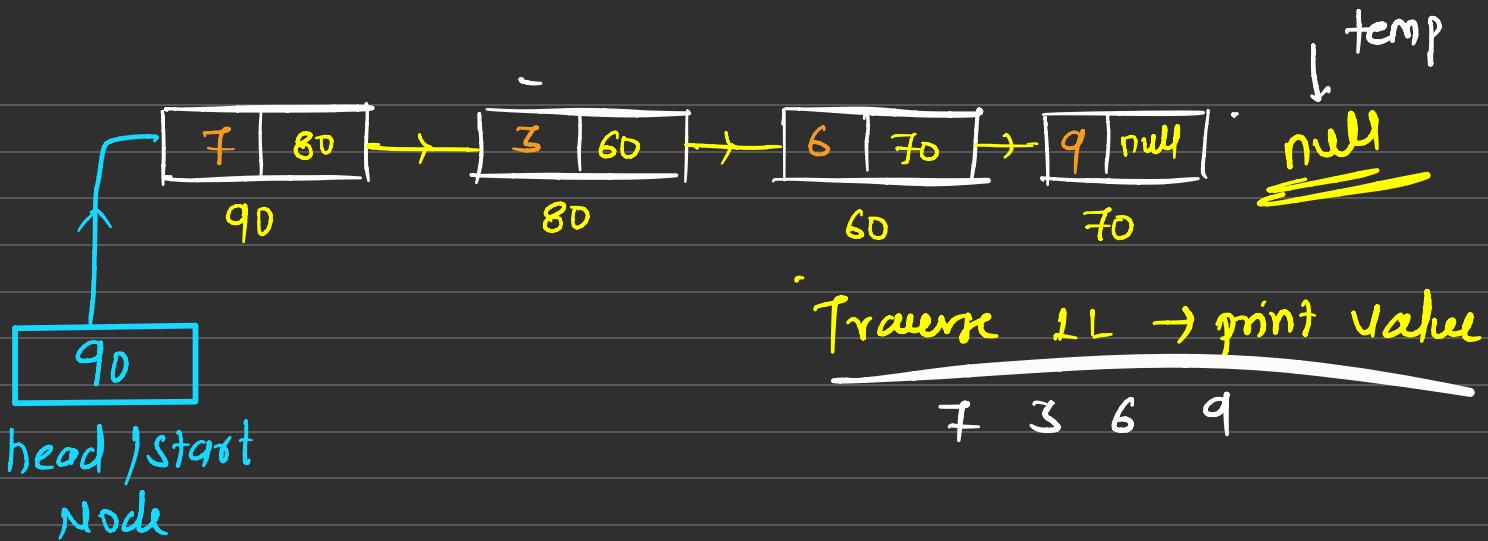
```
class Node  
{  
    int data  
    Node next  
  
    Node (int data)  
    {  
        this.data = data  
        next = null  
    }  
}
```

CPP = Node *node = new Node(10)

Java = Node node = new Node(10)



node
(700)



```

Node * temp = head
while ( temp != null )
    print ( temp -> data )
    temp = temp -> next
}

```

CPP

```

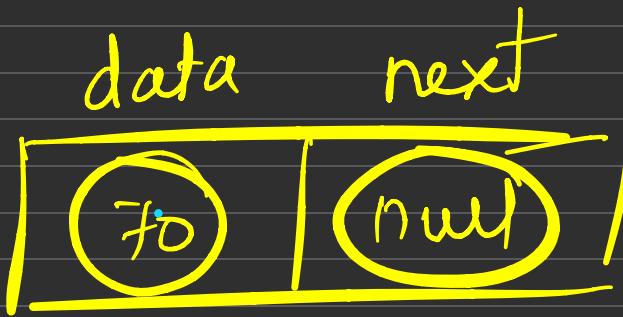
Node temp = head
while ( temp != null )
    print ( temp.data )
    temp = temp.next
}

```

Java

node->next
node->data

(&node).next
(*node).data



node->
node->
node->next
node->data

ताली घजो से पहले तों का
दौर आता है,

ये रिवाज हर कलाकारों
निभाया है !!

Inserction (start)



Node * node = new Node(70)

node → next = head

head = node

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

```
if (head == null)
```

```
}     head = node
```

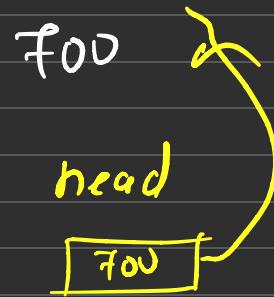
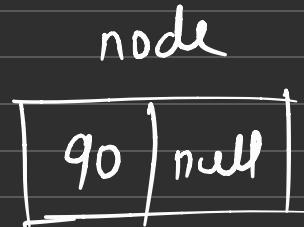
```
}
```

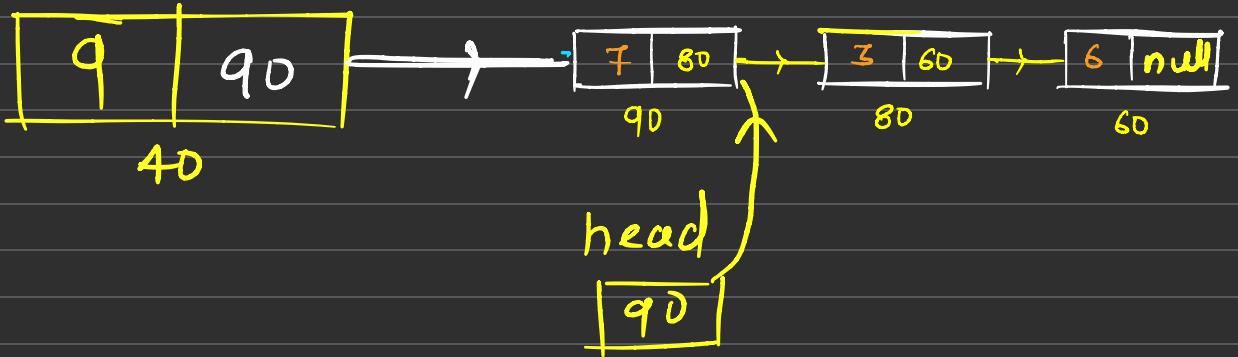
```
else
```

```
}     node->next = head
```

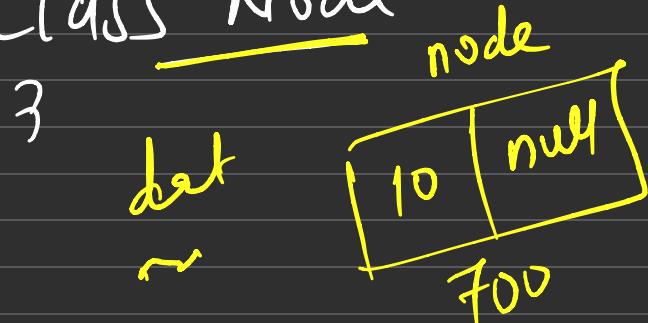
```
        head = node
```

```
}
```



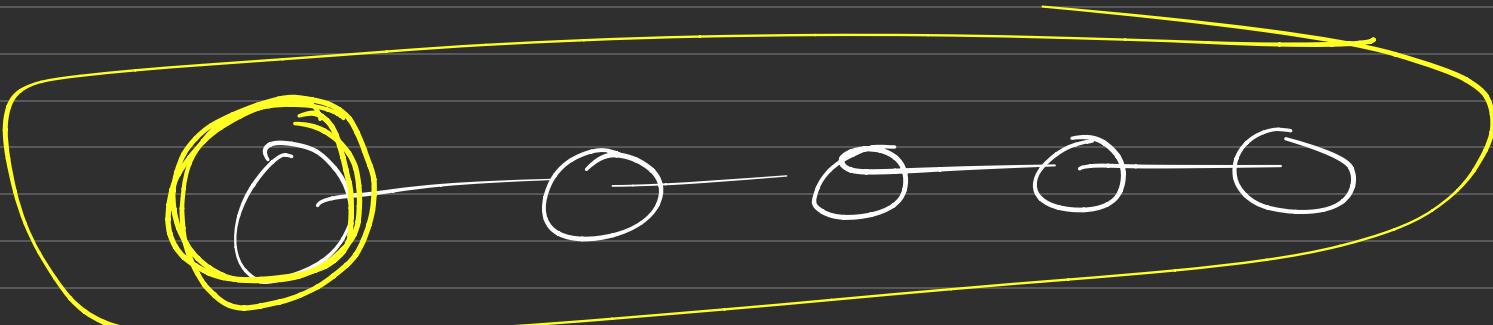
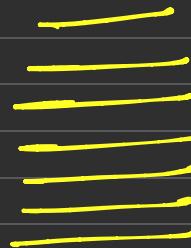


Class Node

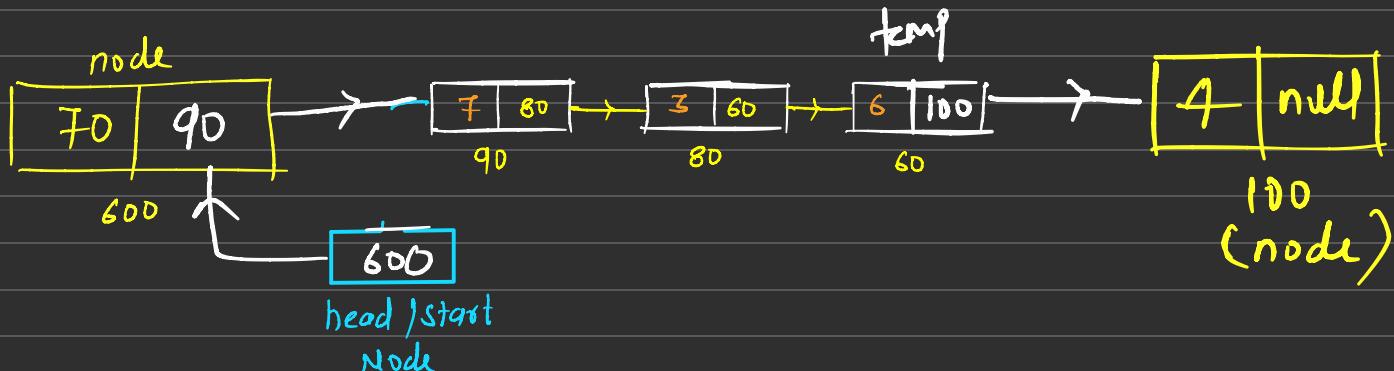


class SLL

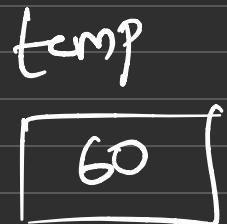
3



Insert (End)



Node * node = new Node(4)



{

if (head == null)

}

↳

head = node

p

K

J

_

else

{

temp = head

while (temp->next != null)

{

temp = temp->next

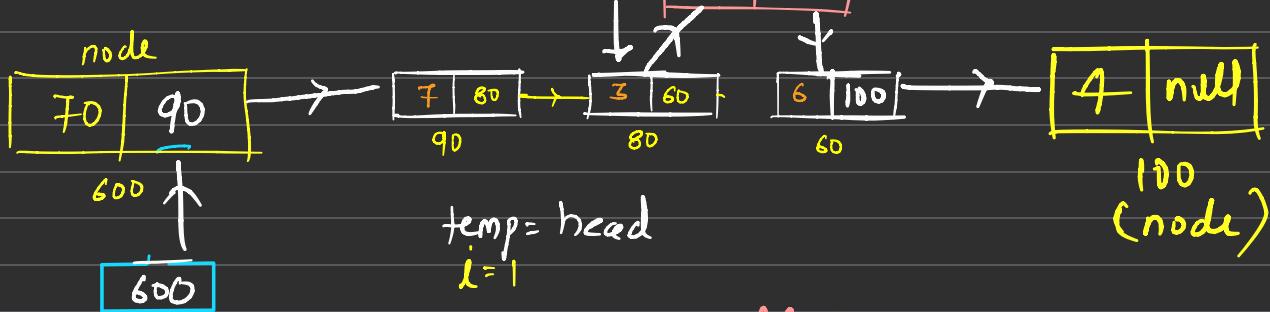
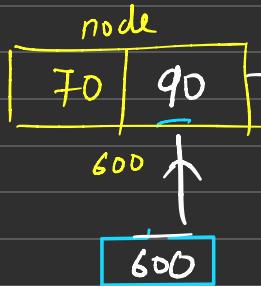
↳

temp->next = node

K

C

Insert (Position)



head) start
Node

while ($\text{temp} \neq \text{null}$ & $i < \text{pos}-1$)

{ $\text{temp} = \text{temp} \rightarrow \text{next}$
 $i++$

}

$\text{node} \rightarrow \text{next} = \text{temp} \rightarrow \text{next}$

$\text{temp} \rightarrow \text{next} = \text{node}$

```
if (pos == 1)  
{ start()
```

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