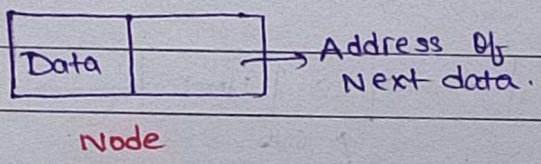
```
int main () {
    Node * first = new Node;
    first → data = 10;
    return 0;
}
```

Lecture - 31

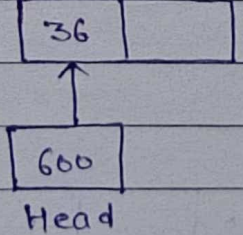
Linked List (Insertion , Delection)




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

Head is always point to first Node.

Collection of Node is known as Linked List



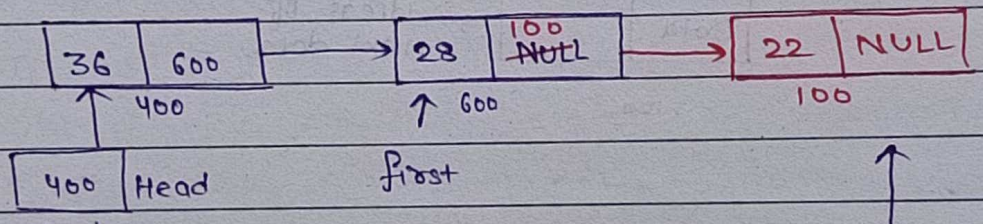
Head can Store address of first Node.

Teacher's Signature.....

Page No.
Date : / /

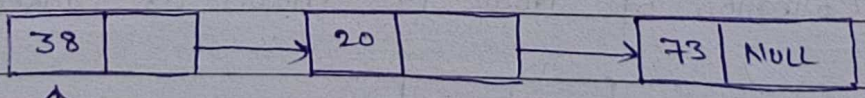
Creation of New Node :

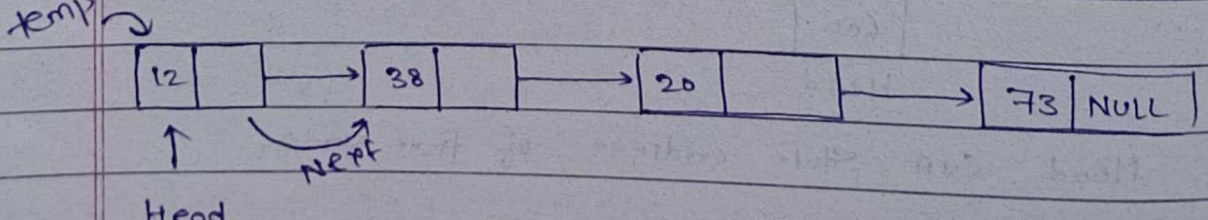
```
int main() {
// 1st Node    Node * Head = new Node();
                Head → data = 36;

// 2nd Node    Head → next = new Node();
                First = Head → next;
                First → data = 28;
```




```
// 3rd Node
                First → next = new Node();
                First = First → Next;
                First → data = 22;
}
```

* Insertion at Beginning :

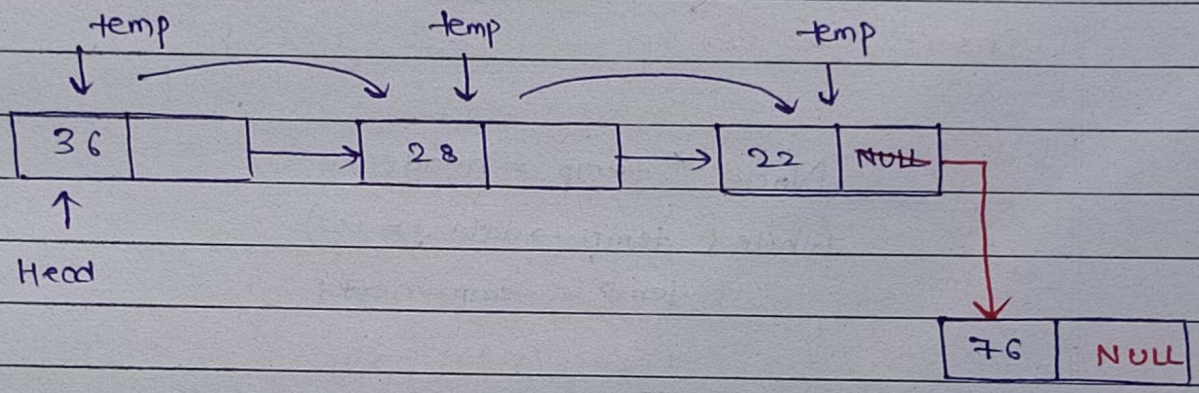




Teacher's Signature.....

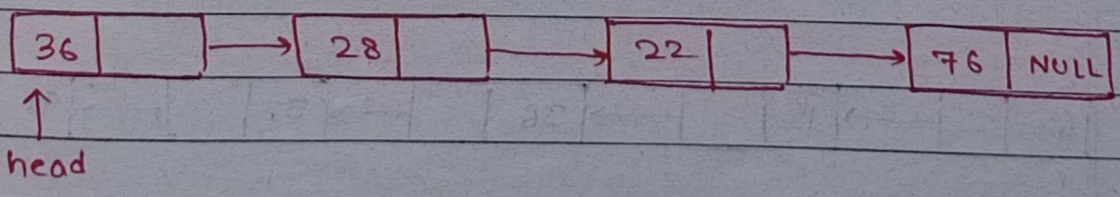
```
Node * temp = new Node();
temp->data = 17;
temp->next = Head;
Head = temp;
```

* Insertion at End :



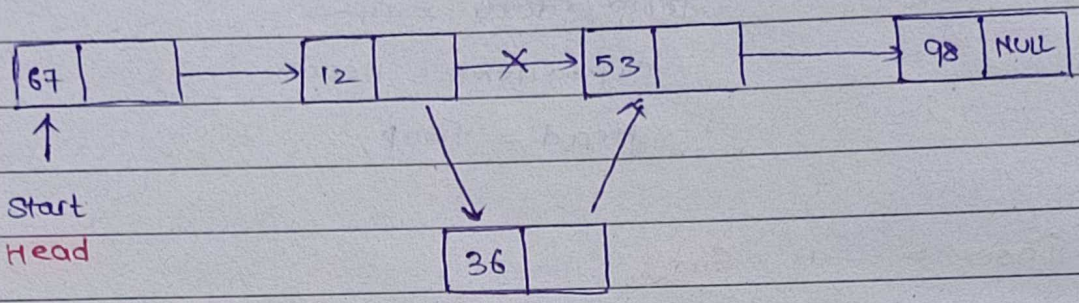

```
Node* temp = head;
While(temp -> next != NULL) {
    temp = temp->next;
}
```

```
temp->next = new Node();
temp = temp->next;
temp->data = 76;
```



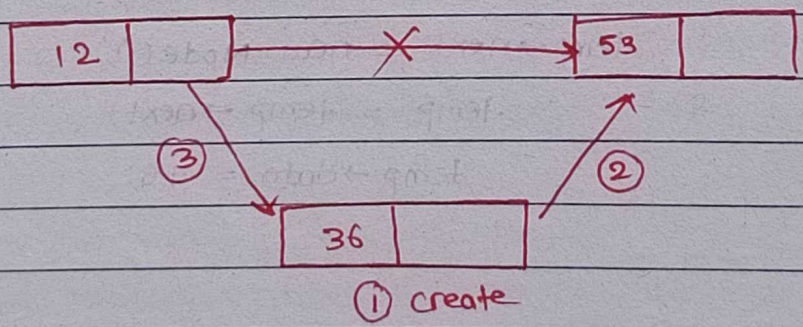
Teacher's Signature.....

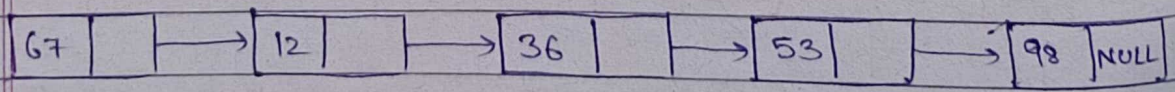
* Insertion At middle (After Node):



Insert After 12.

```
Node * temp = head;
While ( temp -> data != 12)
        temp = temp -> next;

Node* newNode = new Node();
newNode -> next = temp -> next;
temp -> next = newNode;
NewNode -> data = 36
```





Teacher's Signature.....

Page No.:
Date: / /

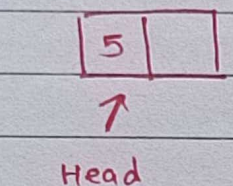
* Code for Converting array into Linked List.

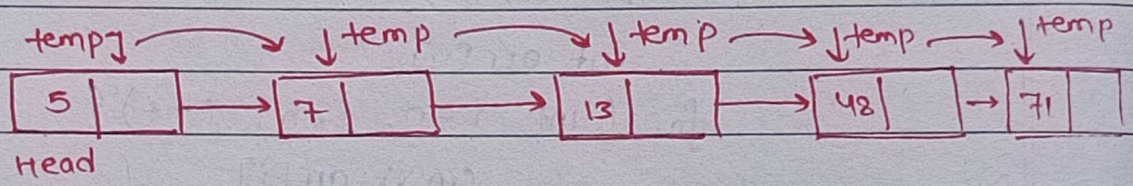
```
Class Node {
    public:
        int data;
        Node * Next;
};

int main () {

    int n;
    int arr[n];
    for ( int i=0; i<n; i++) {
        cin >> arr[i];
    }

Node * head = new Node();
    head → data = arr[0];
    Node *temp = head;
    for (int i=1; i<n; i++ ) {
        temp → next = new Node();
        temp = temp → Next;
        temp → data = arr[i];
    }

    print (head);
    return 0;
}

Void print ( Node * head) {
    While ( head != NULL) {
```

Teacher's Signature.....

```
        cout << head -> data << " ";
        head = head -> next;
    }
}
```

Arr :- 5 7 13 48 71

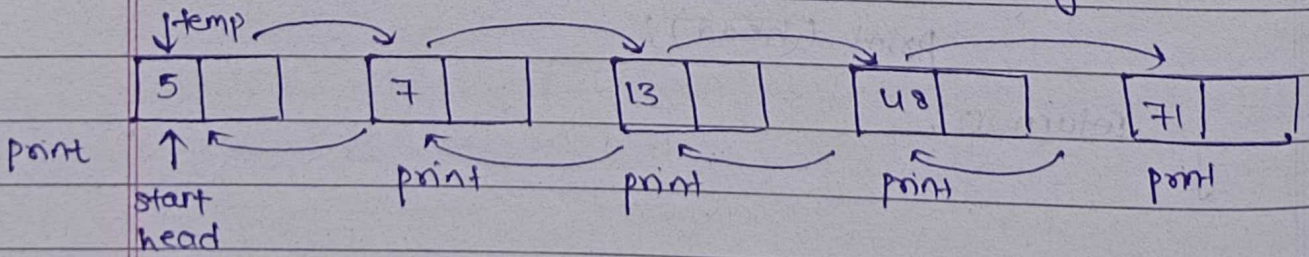




Time Complexity : O(N)

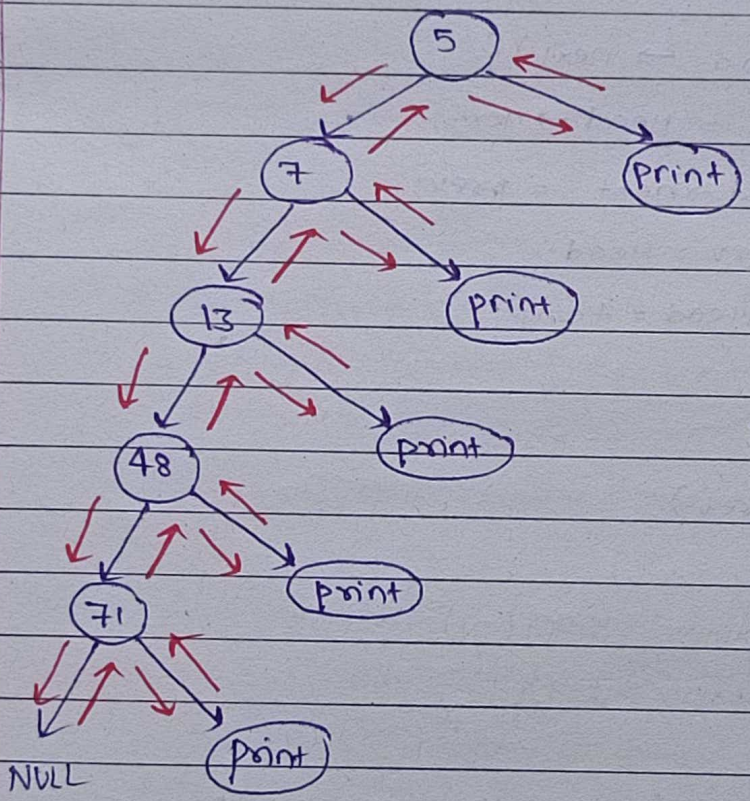
* Reverse of a Linkedlist
  - ↳ By recursion
  - ↳ By Loop

→ By recursion
- ↳ traverse Without printing to the last
- ↳ then back to front With printing.



Teacher's Signature.....

```
reverse (Node * head) {
    if (head == NULL)
        return;
    reverse (head → next);
    Cout << head → data;
}
```

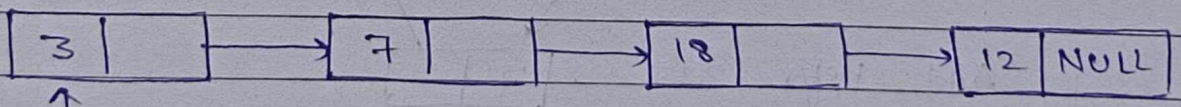


Time Complexity : O(N)

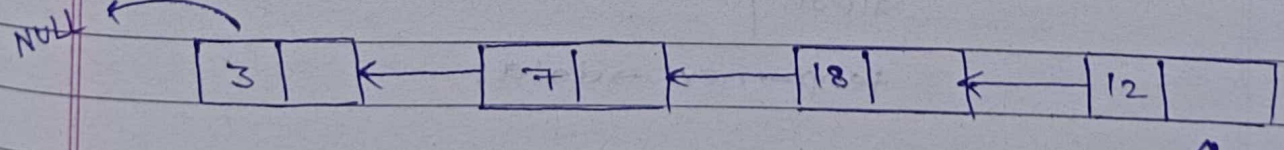
Space Complexity : O(N)

recursion → stack.

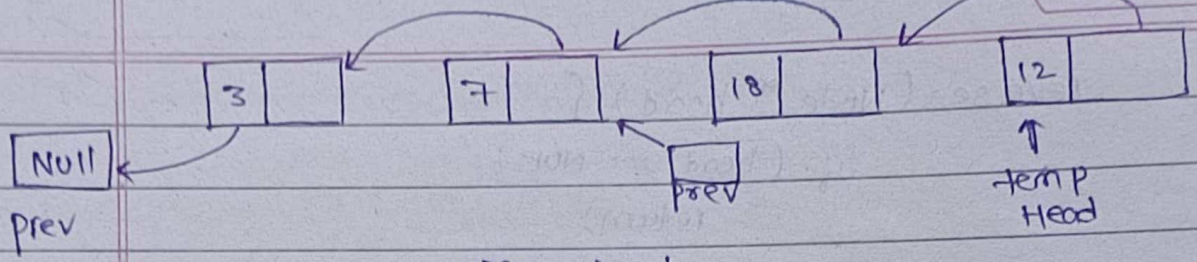
→ By loop :



For. reverse :



Teacher's Signature.....



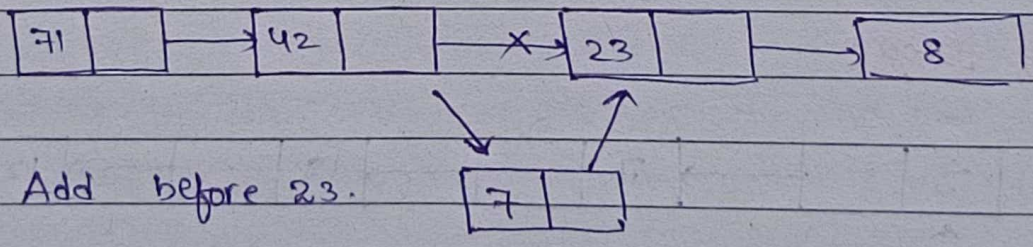
prev = head

```
Node * Prev = NULL;

While ( Head → next ) {
        temp = Head → next;
        Head → next = prev;
        prev = Head;
        Head = temp;
}

print (prev);
```

Time Complexity : O(N)

Space Complexity : O(1)

* Insert in Middle (Before the element):

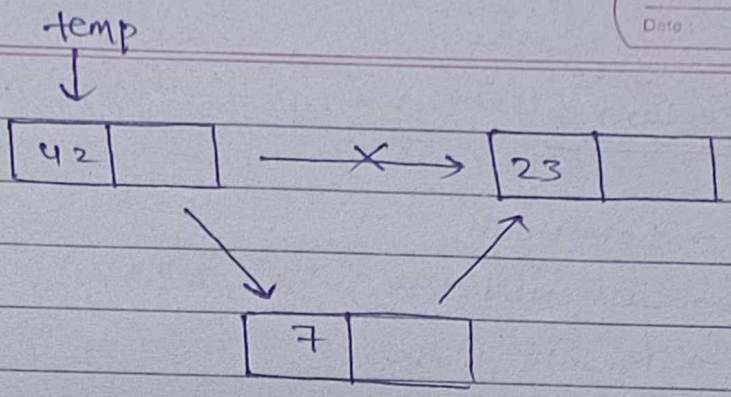


temp → 23 के एक पहले तक

जाएगा।

temp → next → data.

Teacher's Signature.....



Head → next → Data.

```
While ( temp → next → data != 23)
        temp = temp → next;
```

Now, insert the value after temp.

Condition : If 23 is first element

```
if ( Head → data == 23) {
        Add Node at first
        // Write Code.
}
```

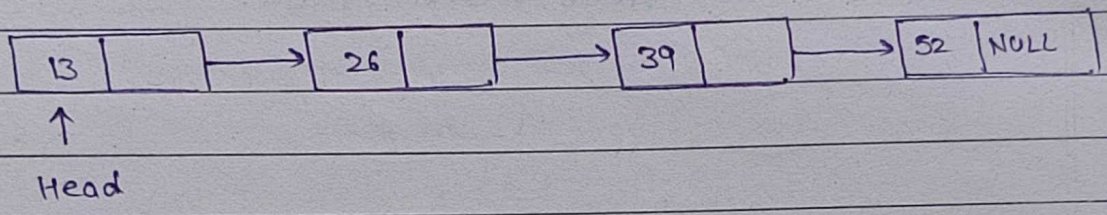
Teacher's Signature.....

Page No:
Date: / /

* Delection :

→ 1st Node
→ Last Node
→ Middle Node

1st Element :

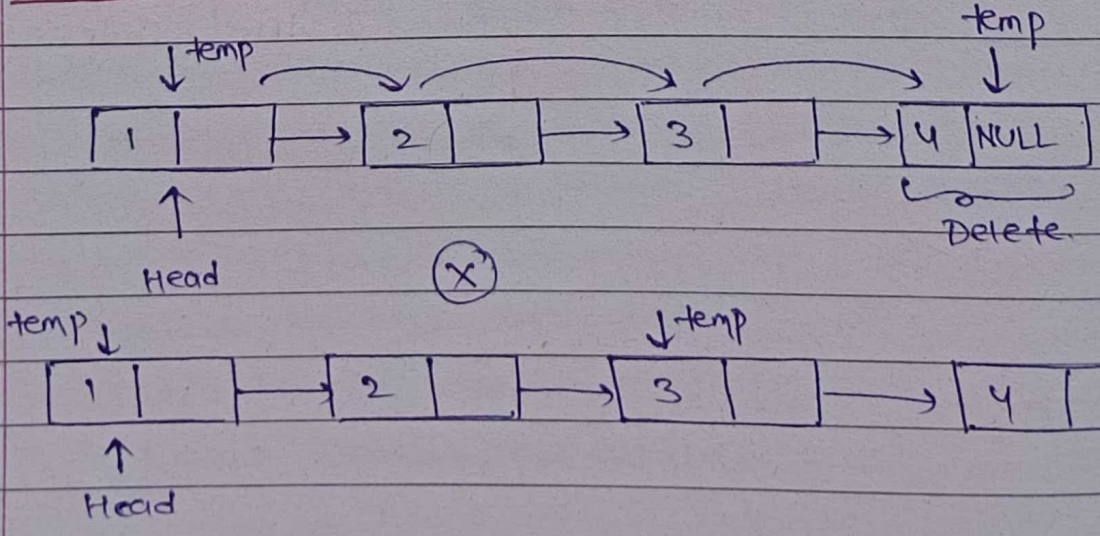


For Delection 1st Element

① Head = Head → next;

② first = Head;
Head = Head → next;
delete (first);

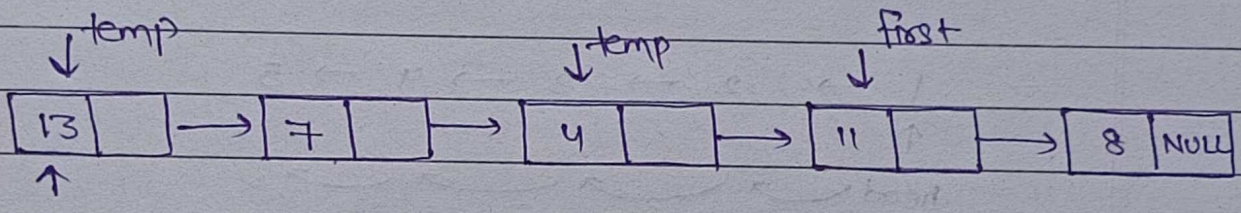
Last Element :



Teacher's Signature.....

```
delete (temp → next)
temp → next = NULL;

While (temp → next → next) {
    temp = temp → next;
}
temp → next = NULL;
delete (temp → next);
temp → next = NULL;
```

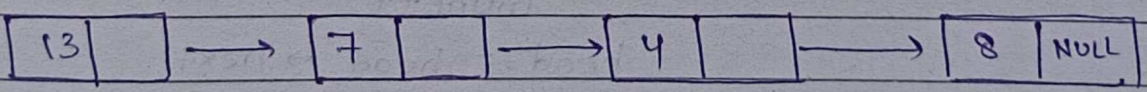
→ Delete the Middle Element :



delete After 4

first

```
first = temp → next;
temp → next = temp → next → next;
delete (first);
```



Teacher's Signature.....

Page No.:
Date: / /

* Count Nodes of Linked list :

A Single linked list

Count Number of Nodes

(1) 1 → 2 → 3 → 4 → 5

traverse the linked list till last
and Count++.

1 → 2 → 3 → 4 → 5
↑
head → → → → →

Count = 1+1 = 2+1 = 3+1 = 4+1 = 5

While (head)
head जब तक NULL नहीं होता।

```
int getCount ( Struct Node * head){
        int Count = 0;
        While (head){
            Count++;
            head = head → next;
        }
    return Count;
}
```

Teacher's Signature.....

Page No.:
Date: / /

* Insert in a Sorted List :

Given a linked list :
in sorted order.

You have given a number. Insert the number at the space where it can be fit.

⇒ 25 → 36 → 47 → 58 → 69 → 80

(19) → Insert at first.

⇒ 50 → 100

(75) → Insert in Middle.

We have to write code for insertion for all the case.

```
class solution {
    public :
    Node * sortedInsert ( struct Node *head, int data) {
        if (head → data > data){
        Struct Node * temp = New Node(data);
            temp → next = head;
            head = temp;
            return head;
        }
```

Teacher's Signature.....

```
struct Node * first = head;

While( first → next && first → next → data < data){
        first = first → next;
    }
    Struct Node * temp = new Node (data);
        temp → next = first → next;
        first → next = temp;
        return head;
    }
};
```

Teacher's Signature.....