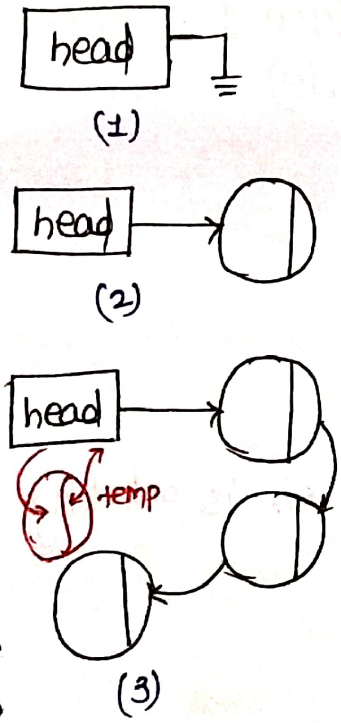
7 important operations of linked list.

- insert • add node at first
- append • add node at last
- insert by pos • add node at specified position
- delete first
- delete last
- delete by pos
- reverse.

1. Insert : New node should be added at 1st position. irrespective of no. of nodes present in linked list.

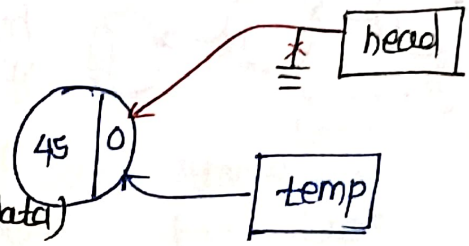
Possibilities:



For insert ⇒

First create node

Node tmp = new Node (data)




```
if (head == null)   // possibility 1
  {
    head = tmp;
  }
```

here head should point added new node.

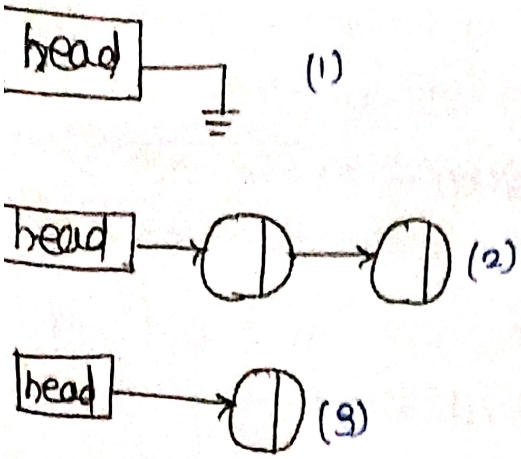
```
else (head ≠ null) // possibity 2 or 3
  {
    ~~head = tmp~~;
```

// but if we do this first, we will lose 1st node address.

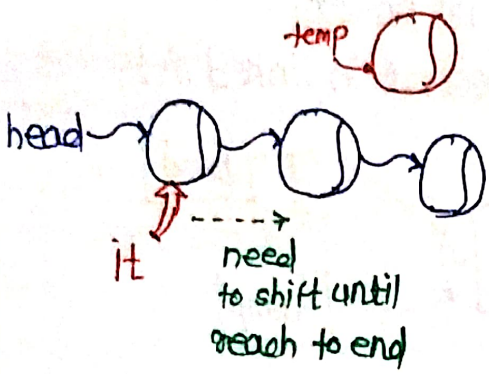
So first copy reff of existing node into added node & then change head

```
    temp.next = head;
    head = temp;
  }
```

2. Append : Insert Node at last.

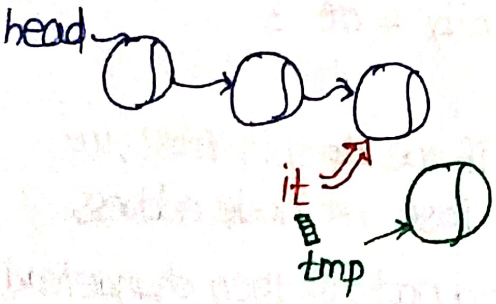



```
public void append(int data)
{
Node temp= new Node (data);
if (head == null); // possibility (1)
{
    head = tmp;
}
else        (2 & 3)
```



// we want to add node at last but we don't hadve a reff of last node. we have to go from start only but we can't shift head as if it go further it will lose head reff so take 1 local reff it (iterator to iterate)

```
Node it = head;
while ( it.next != null
{
    it = it.next;
}
```



// it reached at last node who is pointing null

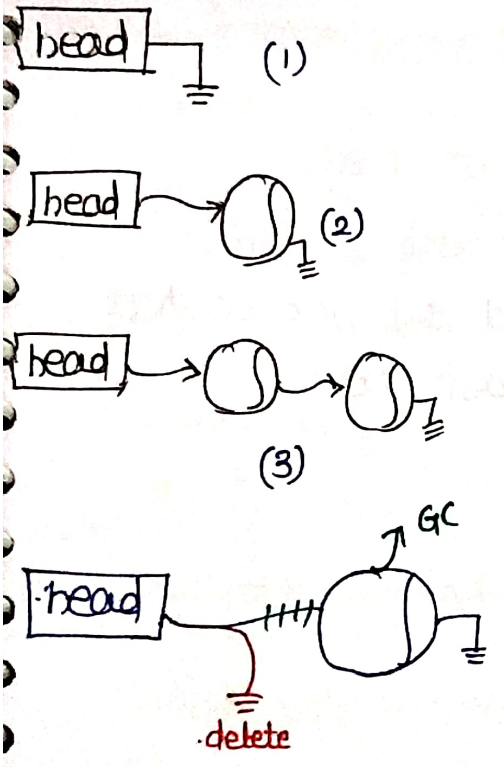
```
it.next = tmp;
tmp.next = null;
}
```

```
public String toString()
{
    string ste = " ";
  if (head == null)
  {
    ste = "LLisempty";
  }
```

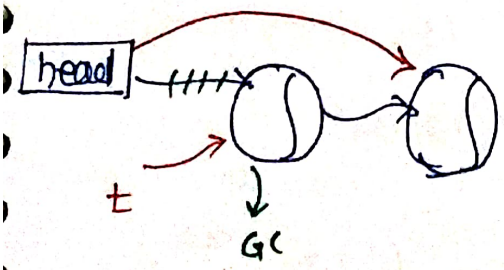
```
else
{ Node it = head;
  while (it.next != null)
  {
    head ste += =
```

```
else
{
    Node it = head;
    // while (it.next != null   ==> for without last node
    while (it != null)          ==> for last node.
    {
        str += it.data + ", ";
        it = it.next;
    }
return str;
}
```

3. Delete first : To delete the first node of linked list.




```
public void deleteFirst ()
{
    if (head==null)   // possibility-1
    {
        sysout ("LL is empty");
    }
else   // check (2) or (3)
    if (head.next ==null) // possibility (2)
    {
        ~~head = null~~; // GC can take whole link
        Node t = head;
        head = t.next;
        t.next = null;
    }
```

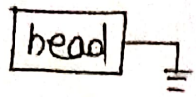


but this GC can take full link as 1st point 2nd so first 1st link to 2nd should break

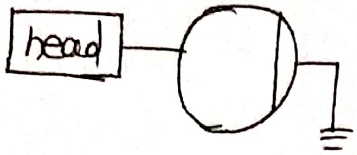
4. Delete last : delete the last node from the linked list.

Possibilities :-

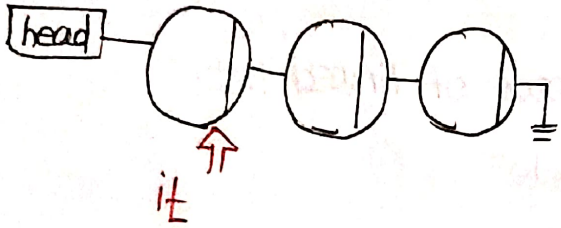
i)



ii)



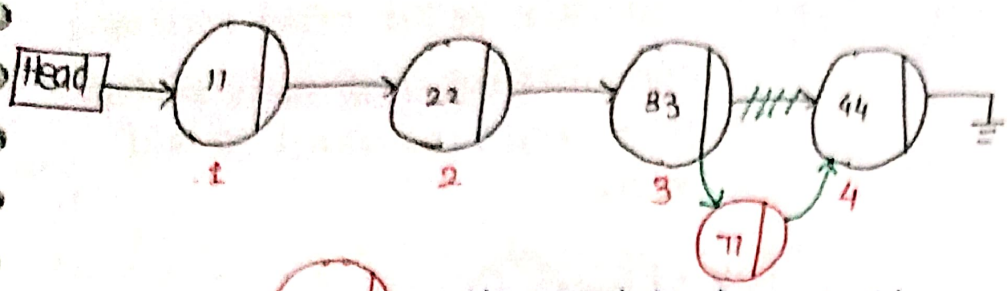
iii)

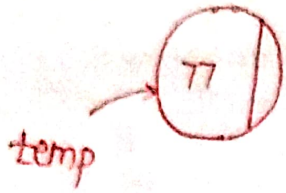



```
public void deleteLast()
{
    if (head == null) // poss-1
    {
        sysout (LL is empty);
    }
    else
    {
        if (head.next == null) // poss-2
        {
            head = null;
        }
        else
        {
            Node it = head;
            while (it.next.next != null)
            // finding second last link so that
            we can dete last one.
            {
                it = it.next;
            }
            it.next = null;
        }
    }
```

April-7.
day-7.

5. Insert by position: Insert node at a specified position.





We want to insert this temp at position no 4. so for that,

3.next should point → temp
temp.next shoud point → 44th node

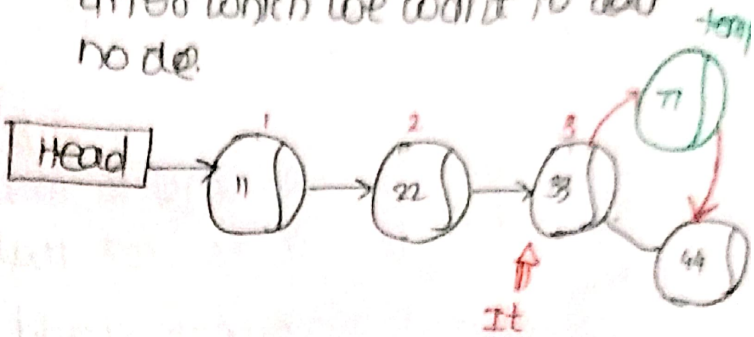
```
Public void insertbypos (int value, intpos)
{   // Find length of LL
    int noOfNodes=0;
    Node it = head;
    while (it !=null)
    {
        noOfNodes ++;
        it = it.next;
    }
    Sysout("no of nodes="+ noOfNodes);
```

Getlength fn return no. of nodes.

getLLLength();

```
public void insert_by_pos (int value, int pos);
{
    int noOfNodes= getLLLength;
    if (pos==1)
    {   insert(value);
    }
    if else ( pos== noOfNodes+1)
    {
        append (value)
    }
```

```
else if ( pos > 1 && pos < noOfNodes+1) // inserting at middle.
{   Node temp = new Node(value)
    Node it = head;
    for ( int i=1 ; i<pos; i++) // iterator point head node and
                                   then iterate upto node before
                                   after which we want to add
                                   node.
{
      it = it.next;
}
 Node temp
   tmp.next = it.next;
   it.next = temp;
}
else
{
  sysout("invalid pos")
}
```

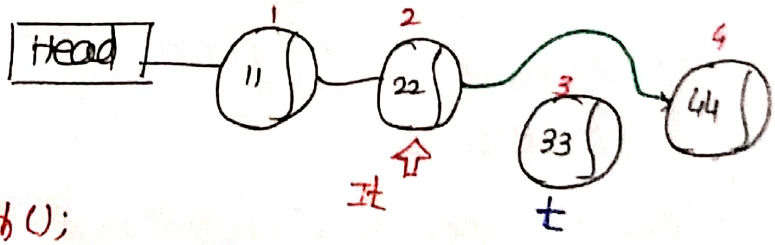


temp.next = it.next

change in temp.next = assign to which it.next is pointing

temp.next में value = it.next जिसको point कर रहा उसकी value.

6. Delete by position: delete the node from a specified position.

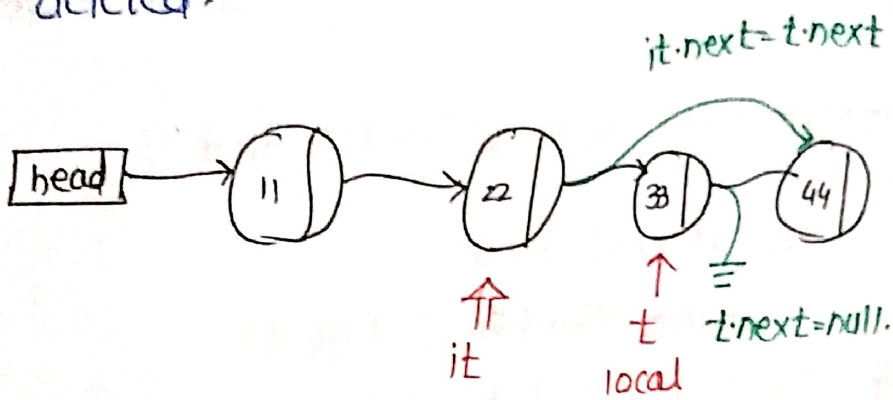



```
public void deletebypos(int pos)
{
 int noNo noOfNodes = getLLength();
 if(pos ==1)
 {
    deleteFirst();
 }
 else if (pos== noOfNodes)
   {
     deleteLast();
   }
```

To delete pos 3, we need to iterate till pos 2.

iterator will iterate till one node before the node which is going to delete.

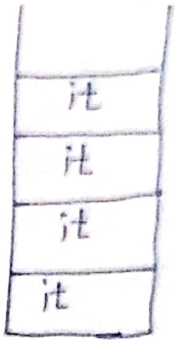
```
else if ( pos > 1 && pos < noOfNodes)
{
    Node it = head;
    For (int i=1 ; i< pos-1 ; i++)
    {  it = it.next;        // reach to 1 pos before the node to be
    }  Node t= it.next        deleted.
       it.next= t.next
    ~~Node t=~~
       t.next = null;
}
else
    {
      sysout ( Invalid pos).
    }
```

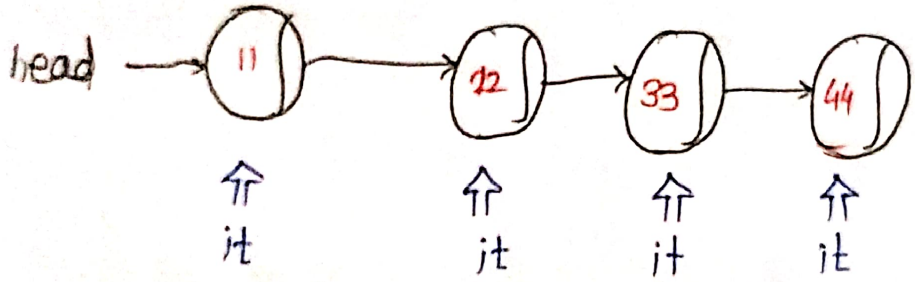


9. Print reverse : to print total linked list in reverse order

```
Public void reversePrint (
{
    reversePrintRec(head);   // wrapper fn : just an entry point
    sysout ("\n") //new line      to recursive fn.
}
```

```
private void reversePrintRec(Node it)
{
    if (it.next != null)
        reversePrintRec ( it.next);
    sysout (it.data + ",");
}
```



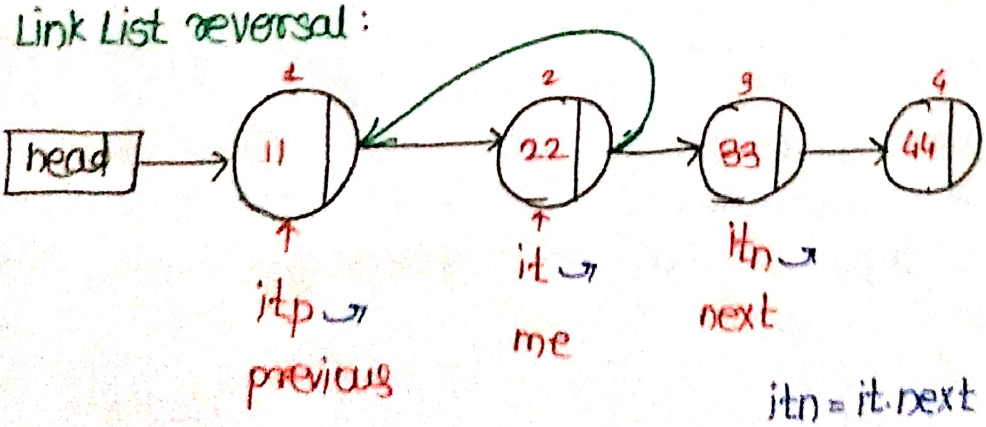


→ Forward (creating it)

← printing reverse

11 22 33 44

Link List reversal :



itn = it.next

it.next = itp // but we can lose next link so,

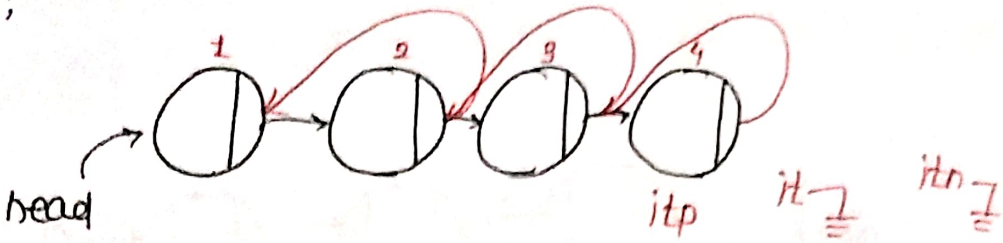
now itp should be shifted to next & same with it & itn

itp = it;

it = itn;

So, the steps to be followed repeatedly are;

```
itn = it.next;
it.next = itp;
itp = it ;
it = itn ;
```



~~ie.~~ ~~while~~

```
public void reverse ()
{
    Node it = head;
    Node itp = null;
    Node itn = head
    while ( it != null)
    {
        itn = it.next;
        it.next = itp;
        ~~it = itn;~~  itp = it;
                       it = itn;
    }
    head = itp ;

}
```

~~list.insert() = ne~~

```
list.insert ( new student ("Ram", 34, 'M');
    {student s1 = new student ("Ram", 34, 'M');
    {list.insert (s1);
```