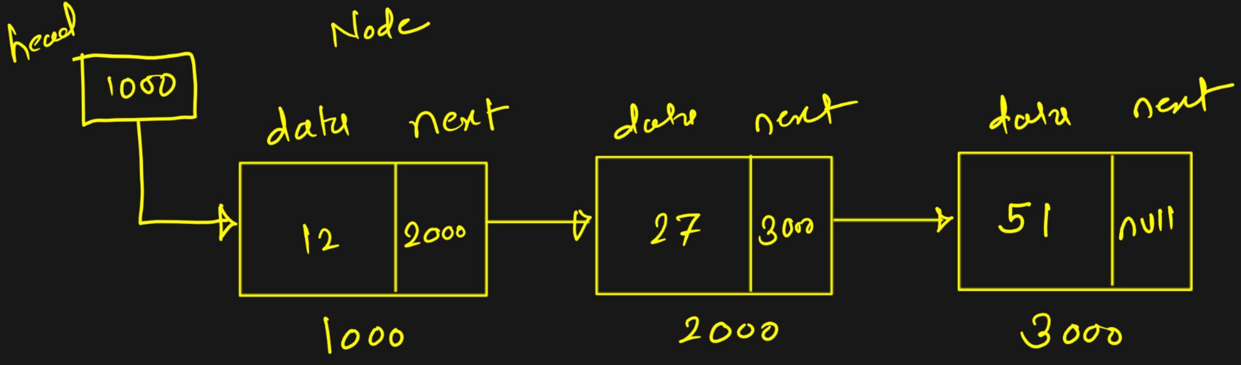
Day-18 Linked List

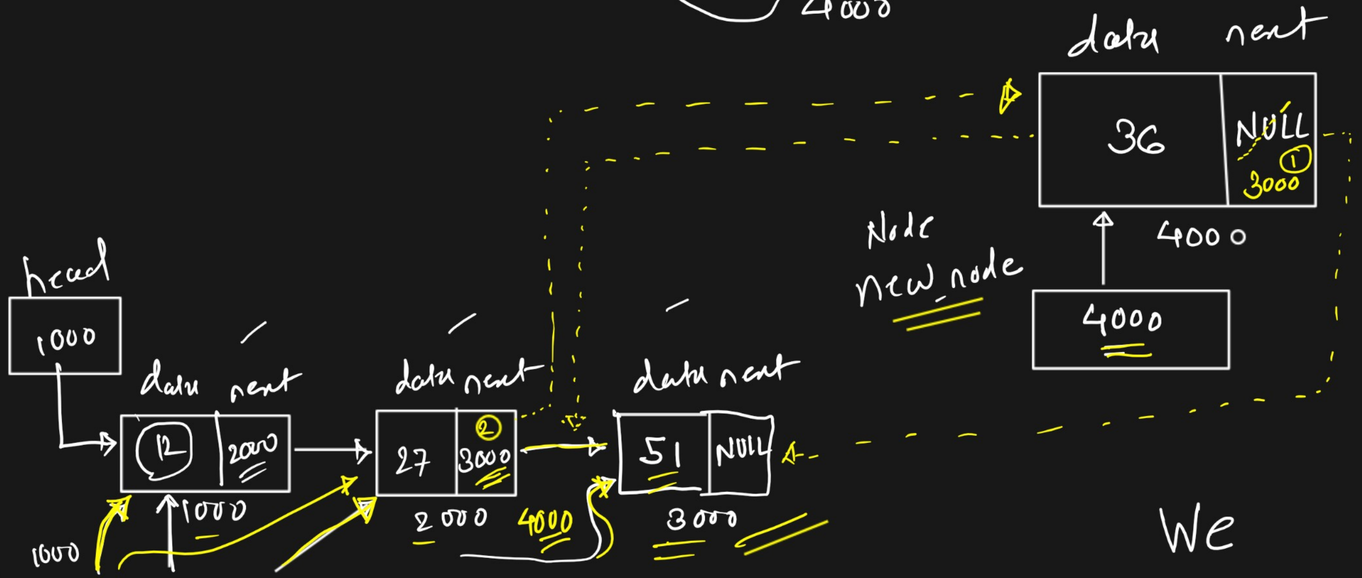


```
Class Node
{
    int data;
    Node next;

    public Node(int num)
    {
        data = num;
        next = NULL;
    }
}

class LinkedList
{
    Node head;

    LinkedList()
    { head = null; }

    public void insert_At_last()
    {
    }

    public void insert_At_First()
    {
    }
```

A) Insert

3) Insert At betn two nodes :-

```
void insert_At_betn(int num)
{
    Node new_node = new Node(num);
                          4000
```



We

$\boxed{\cancel{null}}$ $\boxed{1000}$ ~~2000~~ 3000
prev curr

True 67 True F

```
while ( current.data != posn && current
                                != null)
{
Traverse
    prev = current;          ✓
    current = current.next;
                 3000 NULL
}
```

current.data		posn
12 =	x =	51
27 =	x =	51
51 =	✓ =	51

c	p
1000	0
1000	1000
2000	2000
3000 ✓	

```
if (current == null)
{  s.o.p ("  ______  ");
   return;
}
```

posn x
insert x

```
posn == current.data
new_node.next = prev.next;
                2000 ↓ 300
prev.next = new_node
  3000       4000
```

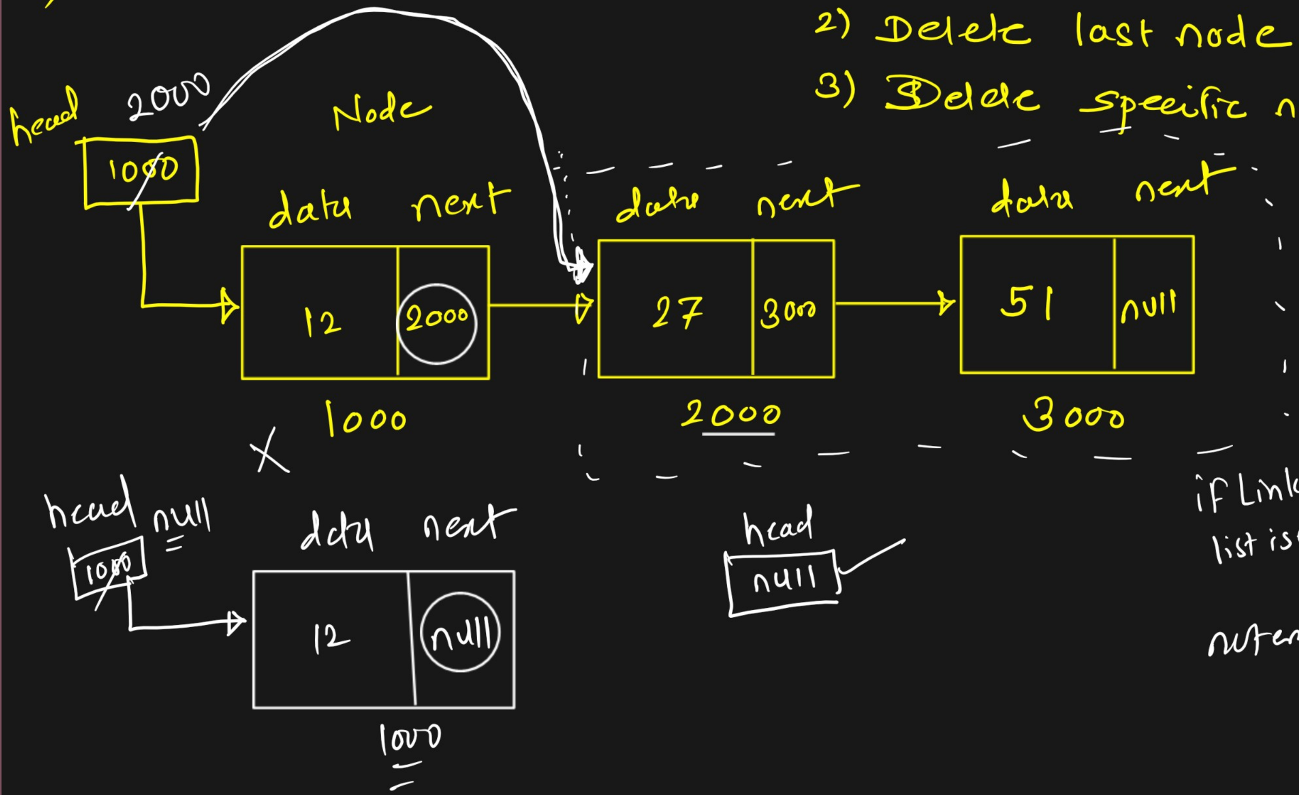
NULL/3000

3000/4000

```
void insert_inbetn (int data,
                    int posn)
{
    if (head == NULL)
    {   s.o.p. ("linked list empty");
        return;
    }

    Node new_node = new Node(36);

    Node prev = null;
    Node curr = head;

    while ( curr.data != posn
    {
```

B) Delete a Node

1) Delete First Node
2) Delete last node
3) Delete specific node



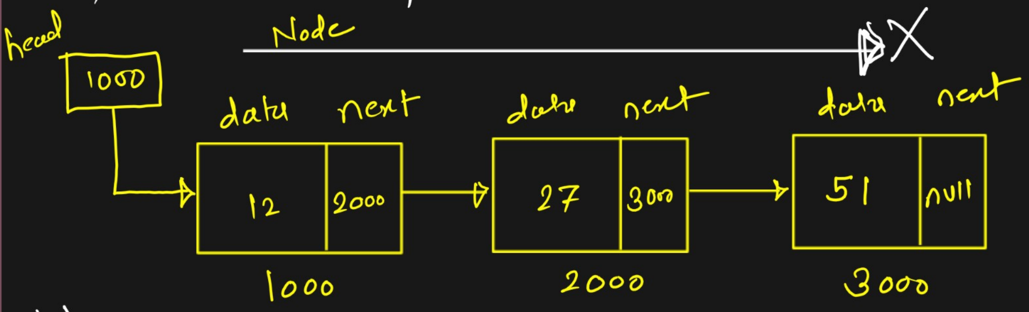
i) Delete First Node

```
void deleteFirstNode( )
{
    if ( head == null)
    {
        s.o.p.("list is empty");
        return;
    }
    head = head.next;
}
```

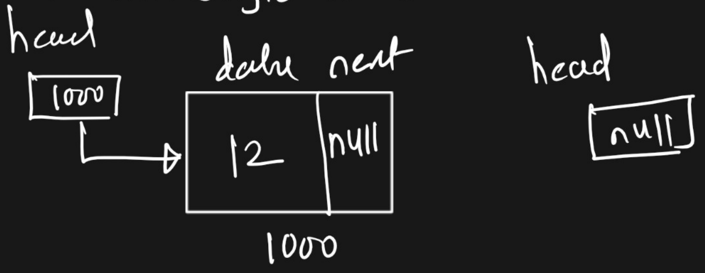
if Linked list is empty

not empty

i) when multiple nodes



ii) when single node

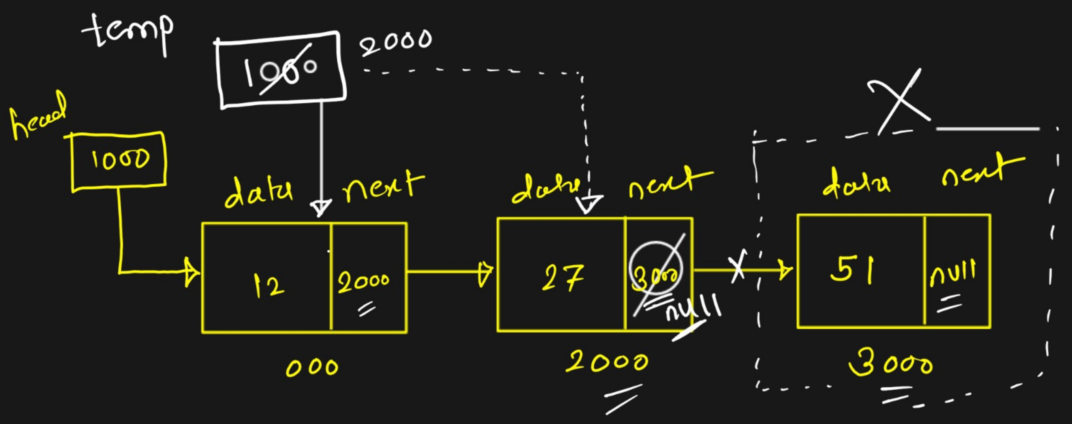


2) Delete Last Node

```
void delete_last_node ( )
{
    if (head == null)
    { s.o.p. ("empty");
      return; }
    else if (head.next == null)
```

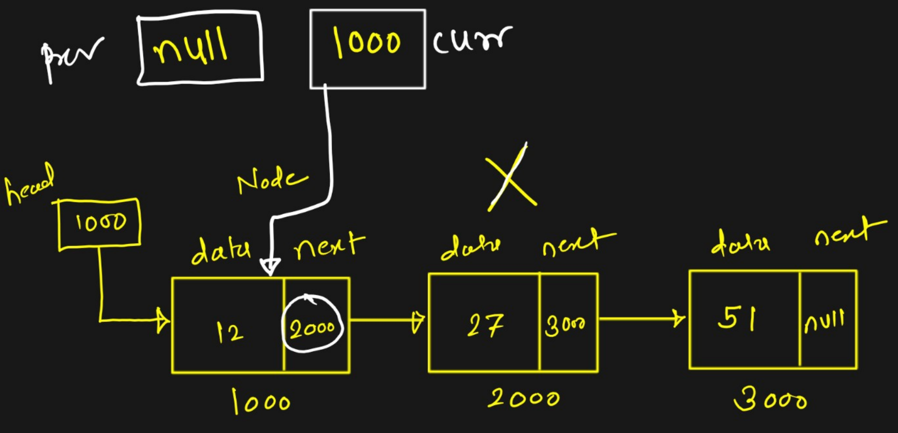
when linked list is empty

for single node

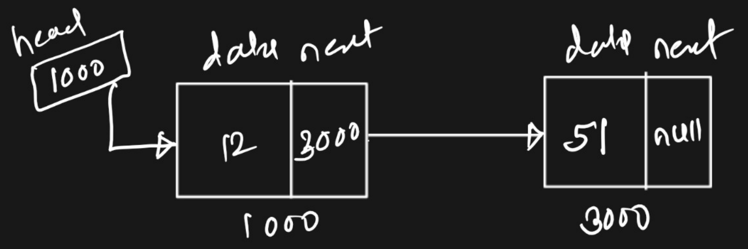


```
{   head = null;  }
    return;
else
{
    Node temp = head;

    while( temp.next.next != null)
    {
        temp = temp.next;
    }

    temp.next = null;
}
}
```

i) temp	temp.next	temp.next.next	
1000	2000	3000	!= null ✓ {
~~2000~~	3000	null	!= null ✗ }

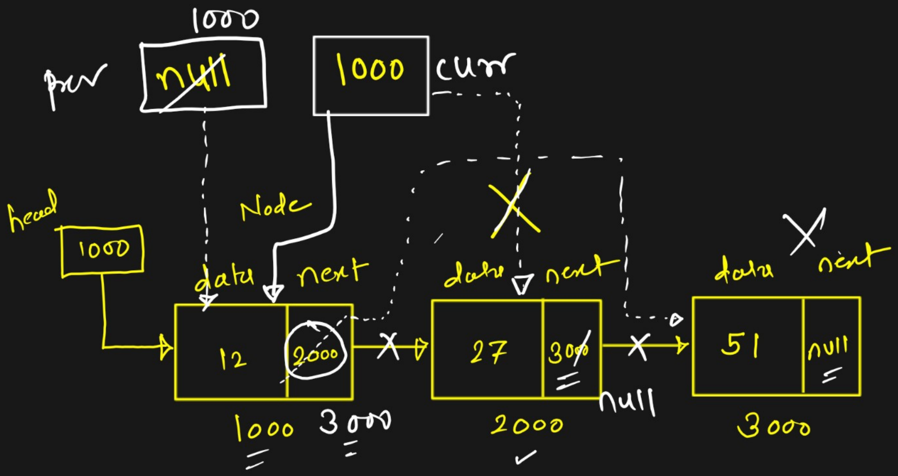


Result ⇒



iii) Delete Specific Node

```
                                      27
void delete_specific_node(int posn)
{
    if( head == null)
    { s.o.p( empty ); return; }
```



prev	curr	curr.next	current.data	pos^n	while cond^n
null / 1000	1000 / 2000	2000	12 = ✗ =	~~27~~ 12	~~true~~ F
1000	2000	3000	27 = ✓ =	27	false
2000	3000	null	51 = ✓ =	51	false

```
Node prev = null;
 Node curr = head;

while (curr.data != pos^n && 
                    curr != null)
{
    prev = curr;
    curr = curr.next;
}

if (curr == null)
{
    S.o.p. ("pos^n does not find");
    return;
}
prev.next = curr.next;
}
```