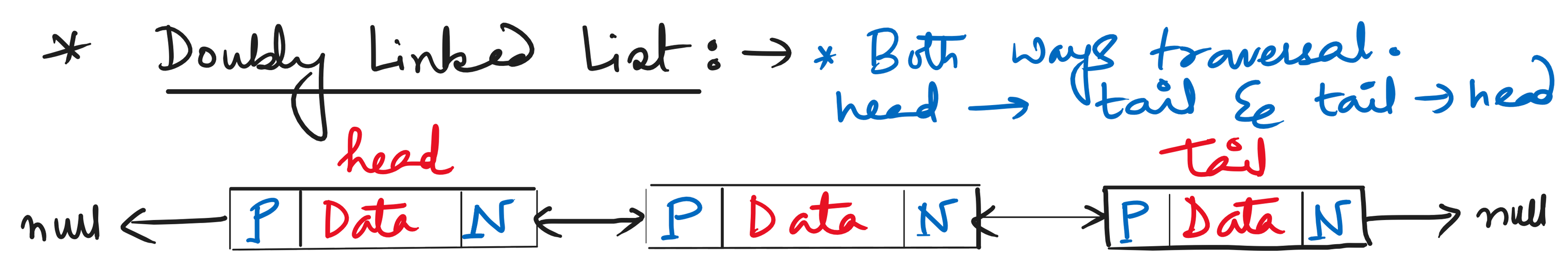
* **Doubly Linked List:** → * Both ways traversal.
head → tail & tail → head

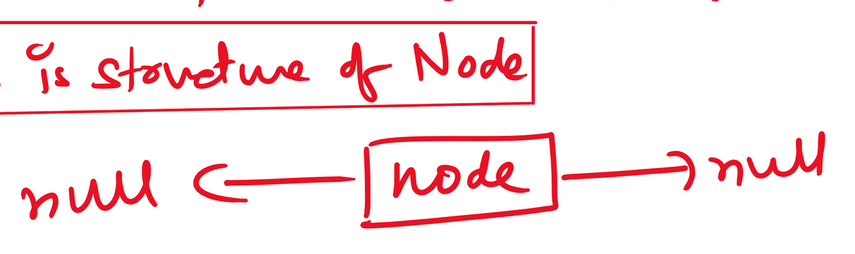


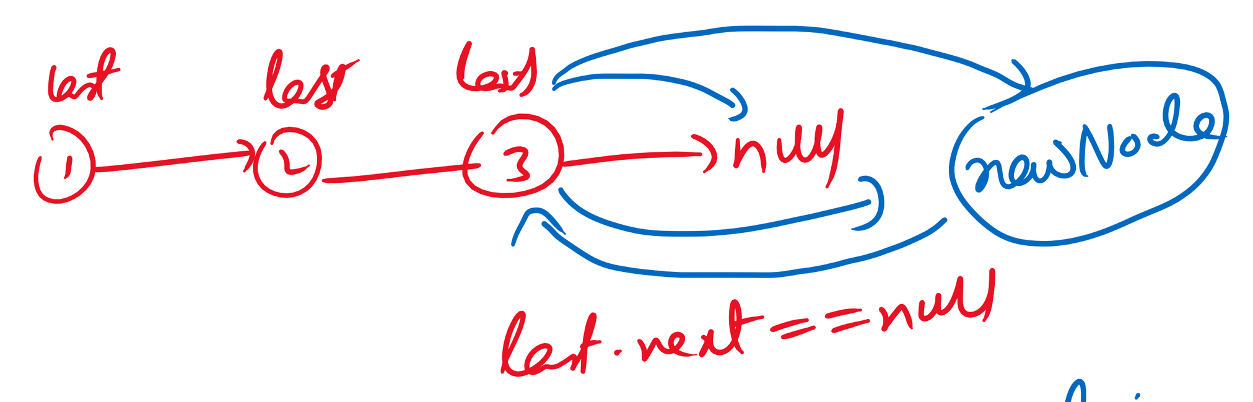
* Each node has two pointers & some data.
* A pointer to the next & a pointer to the previous.
* The previous node of head is "null".
* The next node of tail is "null".

head.prev = null;
tail.next = null;

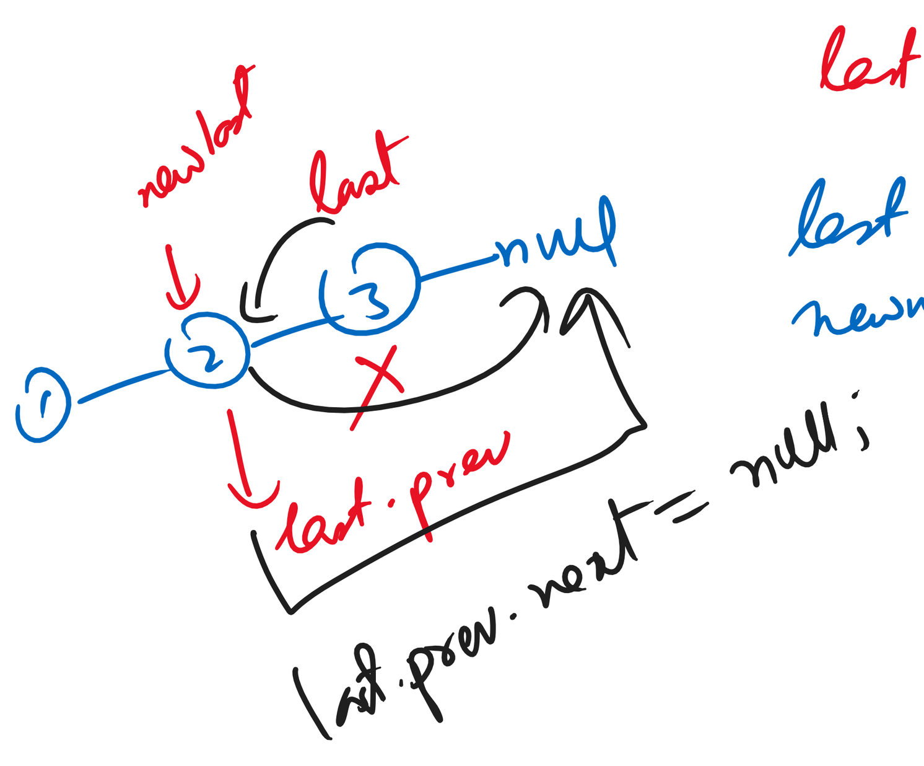
```
class Node {
  int data;
  Node next;
  Node prev;
}
```

This is structure of Node





last.next == null



last.next = newNode;
newnode.prev = last;

last.prev.next = null;

Insert at position: →

(I/P) null ↔ 1 ↔ 2 ↔ 3 ↔ 4 → null
(1st, 2nd, 3rd, 4th)

insertAtPos (int data, int pos)

d 8 , p 3

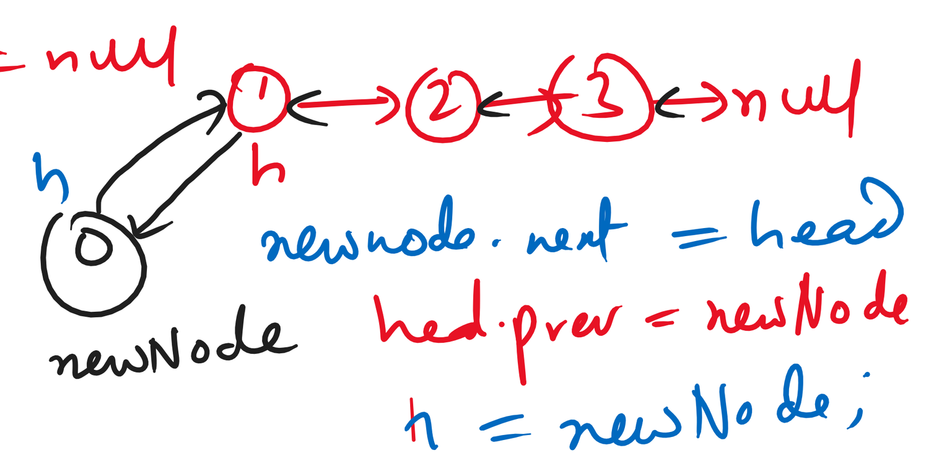
O/P null ↔ 1 ↔ 2 ↔ 8 ↔ 3 ↔ 4 → null
(1st, 2nd, 3rd, 4th, 5th)

[[pos != 0, pos < 0] Not possible] [pos >= 1] Condition

1. Empty h == null
newnode.next = head
newNode → null

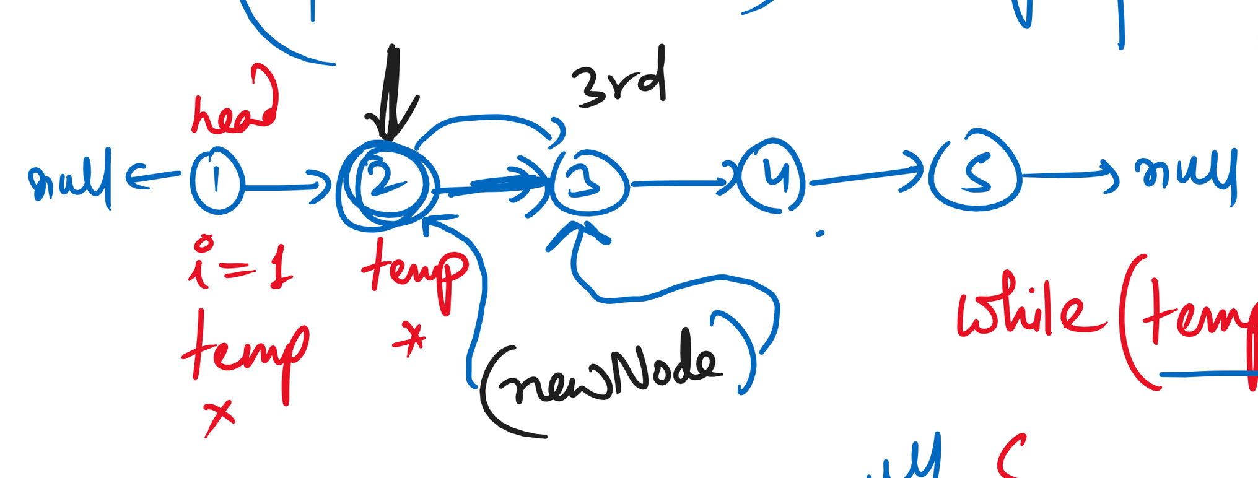
n. Multiple nodes

(pos == 1) head != null



newnode.next = head
head.prev = newNode
h = newNode;

(position != 1) say position = 3 (pos = 9)

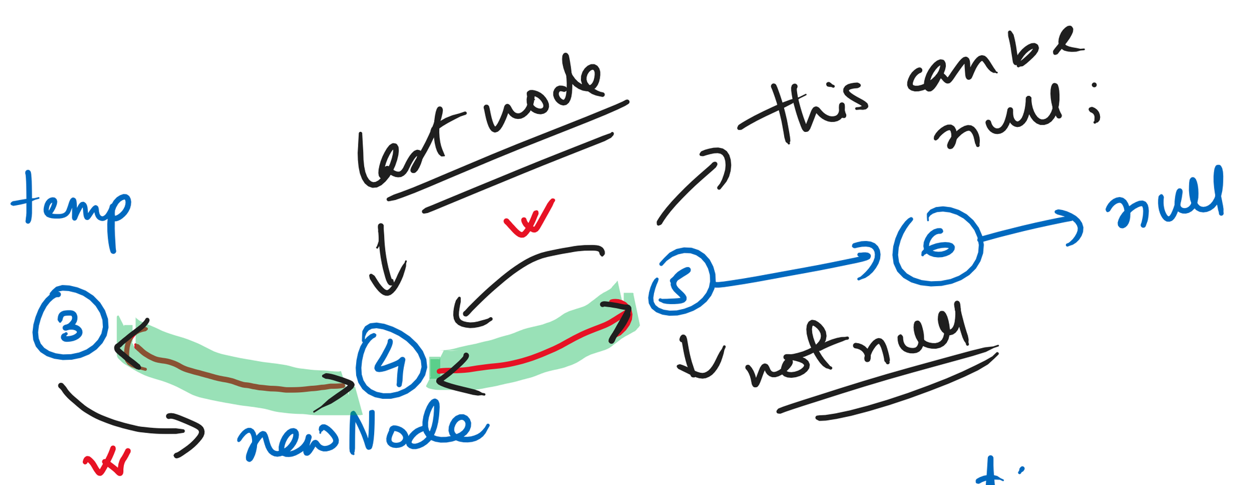



```
Node newnode = new Node(data);
Node temp = head;
while (temp != null &&
       i < position - 1)
{
  temp = temp.next;
  i++;
}
[ newNode.next = temp.next
  newNode.prev = temp. ]
```

temp = null

i = 1, 2, 3

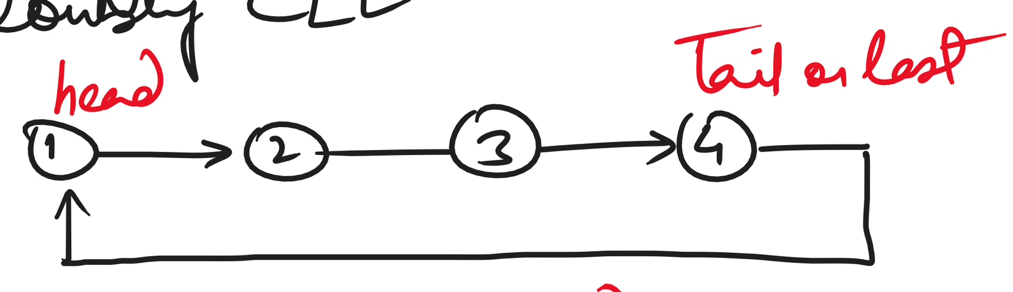
(3<2)
3 < 3 - 1 (3) stop
3<3 stop



newnode.next = temp.next;
newnode.prev = temp;

* In a doubly linked list please make sure you check the connections both ways node.next & node.prev.

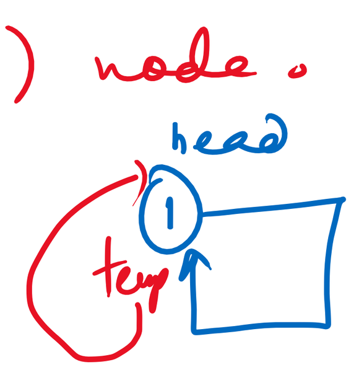
* **Circular Linked List:** →
↳ Singly CLL
↳ Doubly CLL



Tail.next = head

O/P 1 → 2 → 3 → 4 → head

*** The last node is connected back to the first (head) node.



1 → head

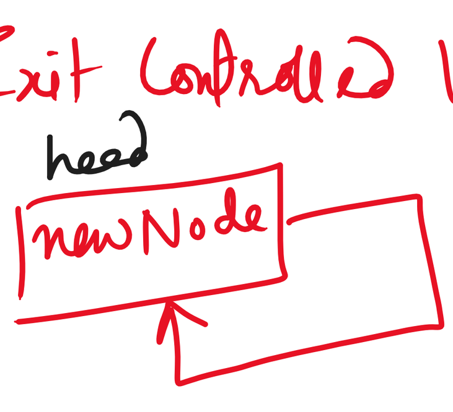
At least once!

do

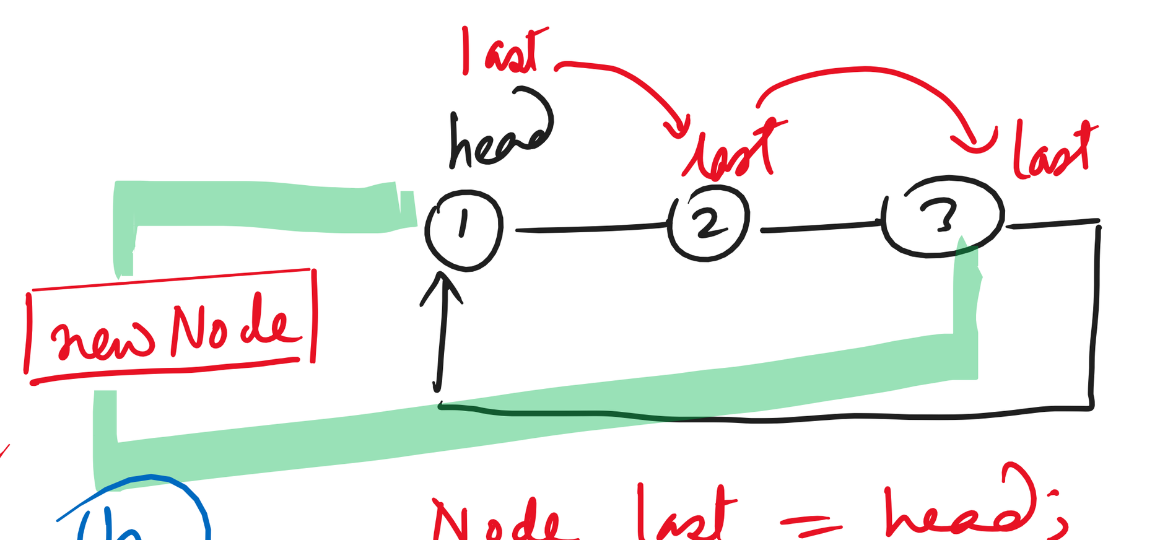
while

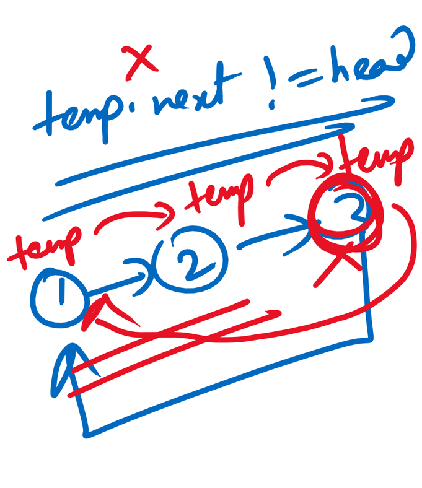
Exit Controlled Loop

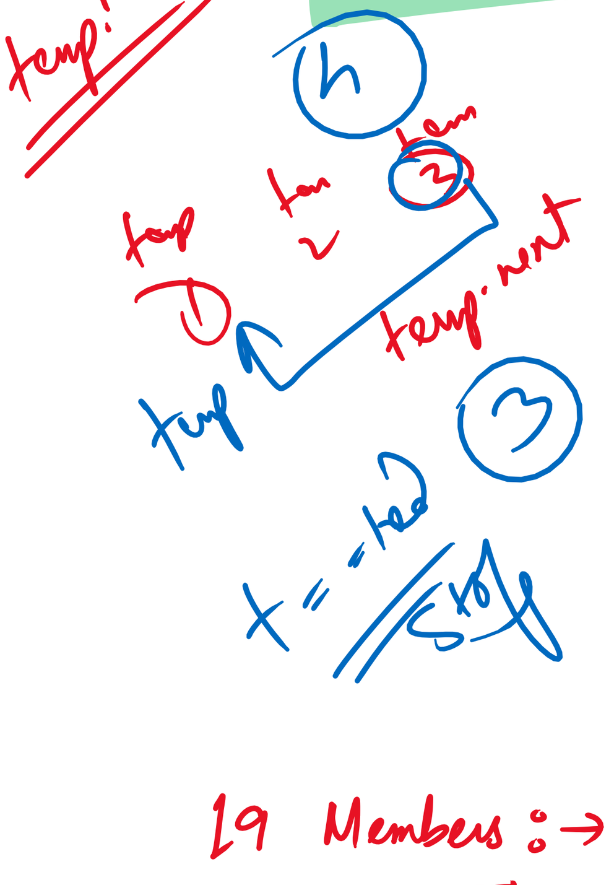
```
Node temp = head;
× while (temp.next != head)
    print (temp.data);
```



nn.n = nn
h = nn








```
Node last = head;
While (last.next != head) {
    last = last.next;
}
last.next = newNode;
newNode.next = head;
h = newNode;
```

19 Members: →