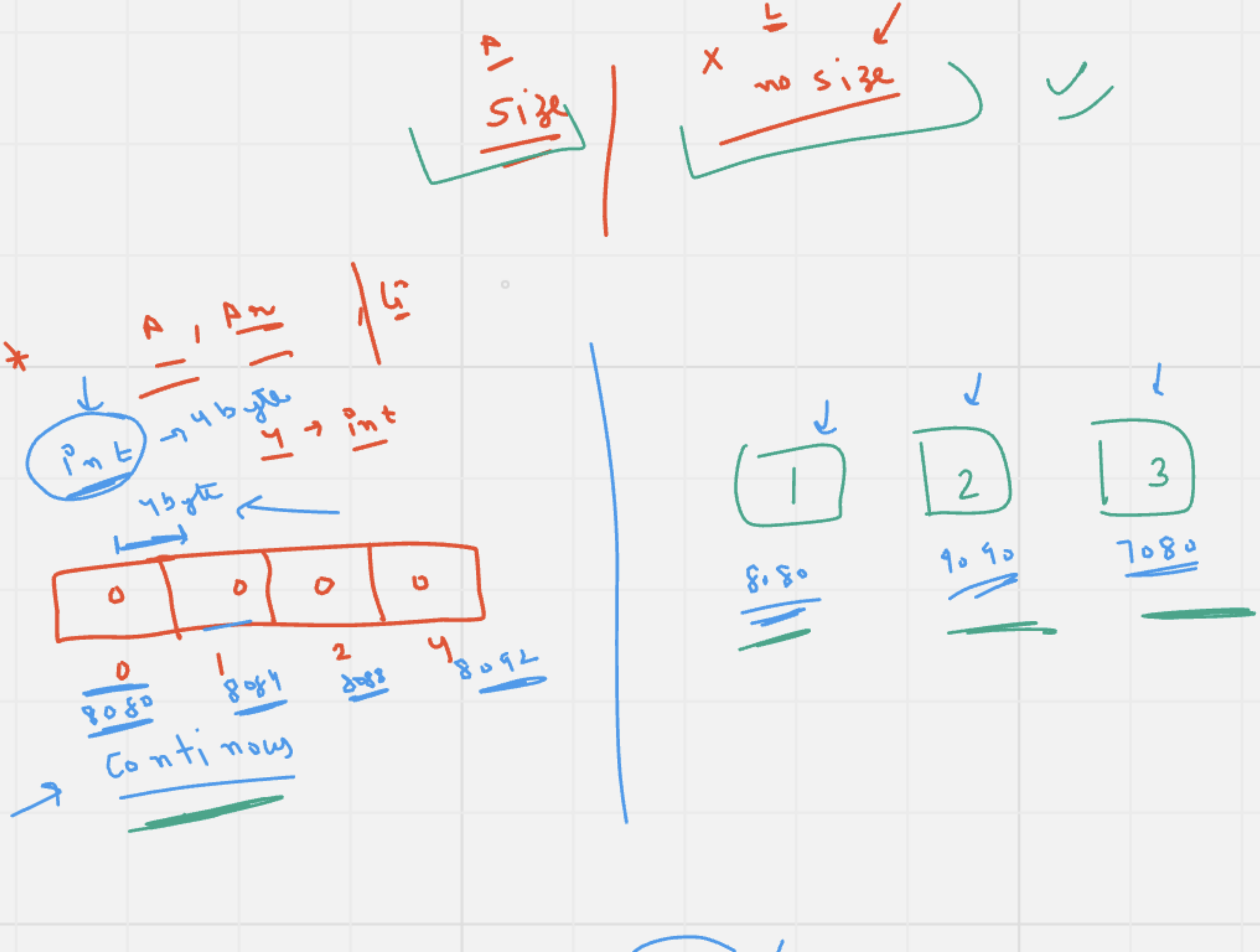
LinkedList


A
size
L
X
no size
*
A , Ar
Li
int
→ 4 byte
4 → int
4 byte
0
0
0
0
0
1
2
4
8080
8084
8088
8092
Continous
1
2
3
8080
9090
7080

*

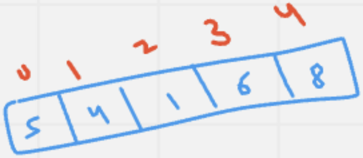
Arr

Size

[10]

arr, list

inc | dec

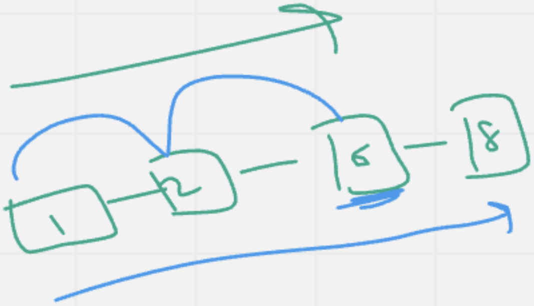


3

arr[3] = 1

O(1)

arr(4) = 8



4 = 8

6

O(N)

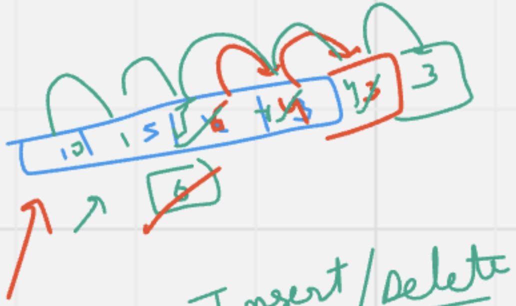
O(1)

ar(4) = 8

start
middle O(N)

End O(1)

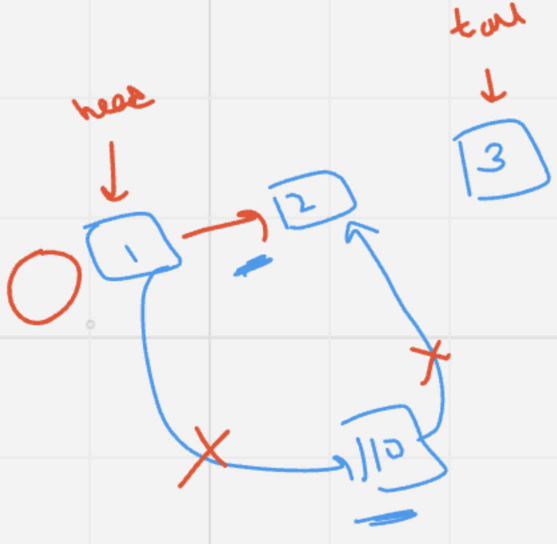
deletion



10

Insert/Delete

O(N)

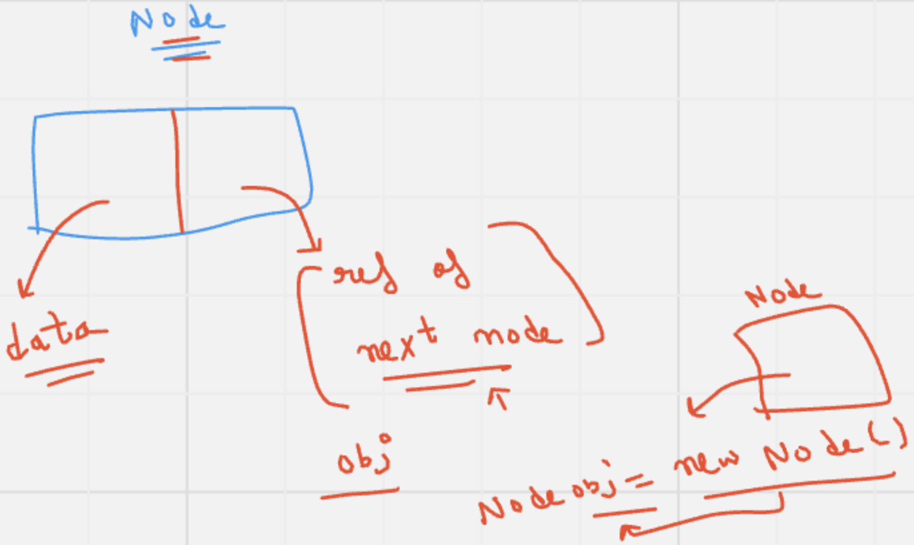


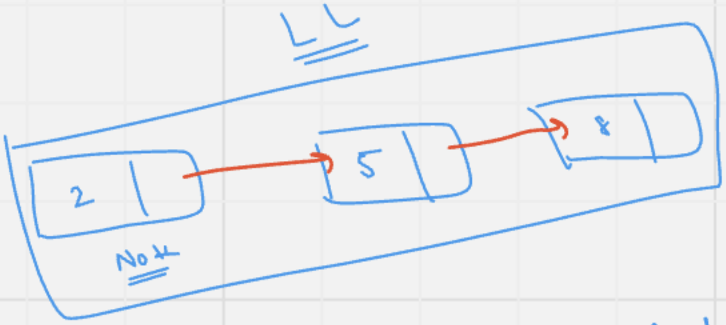
Insert

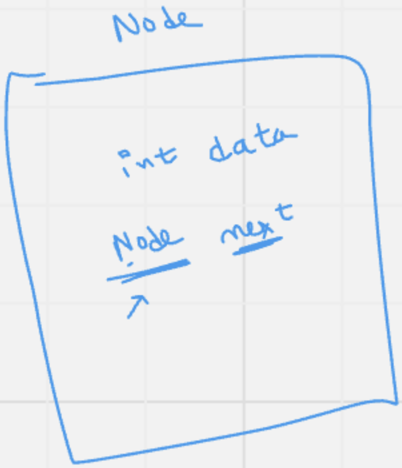
at start
at end O(1)

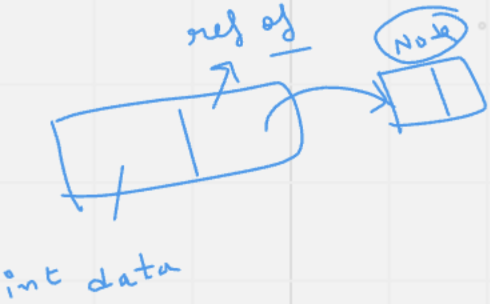
in middle O(n)

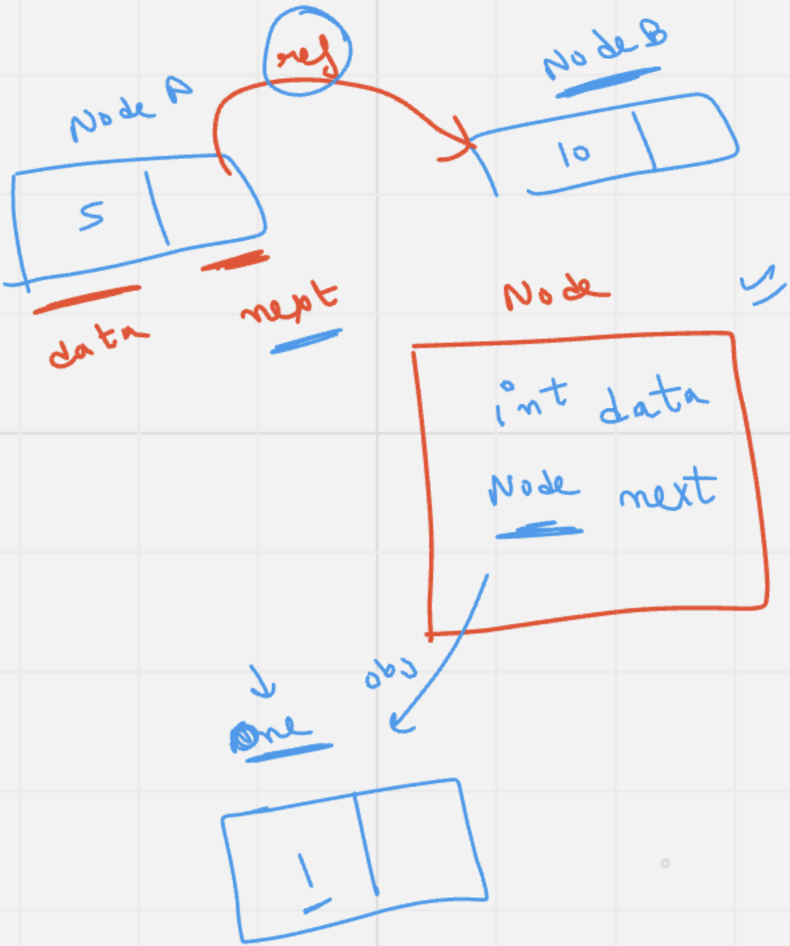
Delet


Node
data
ref of
next node
obj
Node
Node obj = new Node()

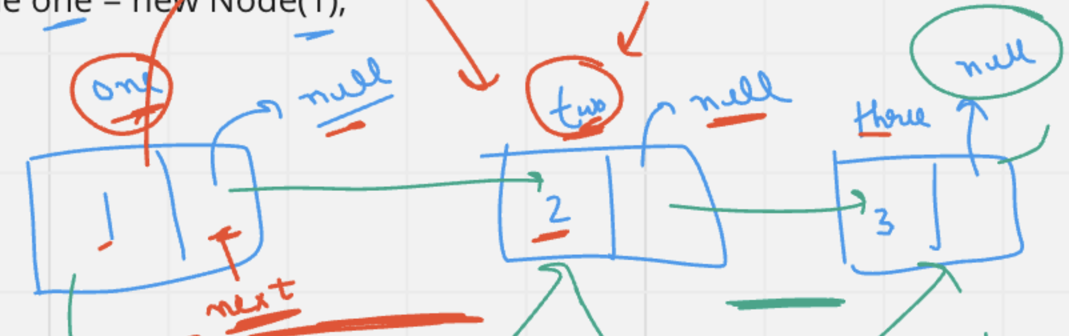

LL
2
5
8
Node


Node
int data
Node next


ref of
Node
int data


ref
Node A
Node B
5
10
data
next
Node
int data
Node next
obj
One
1

// Node one = new Node(1);



two.next = three

two.data → 2

two.next → null

one.next = two

one.next.data

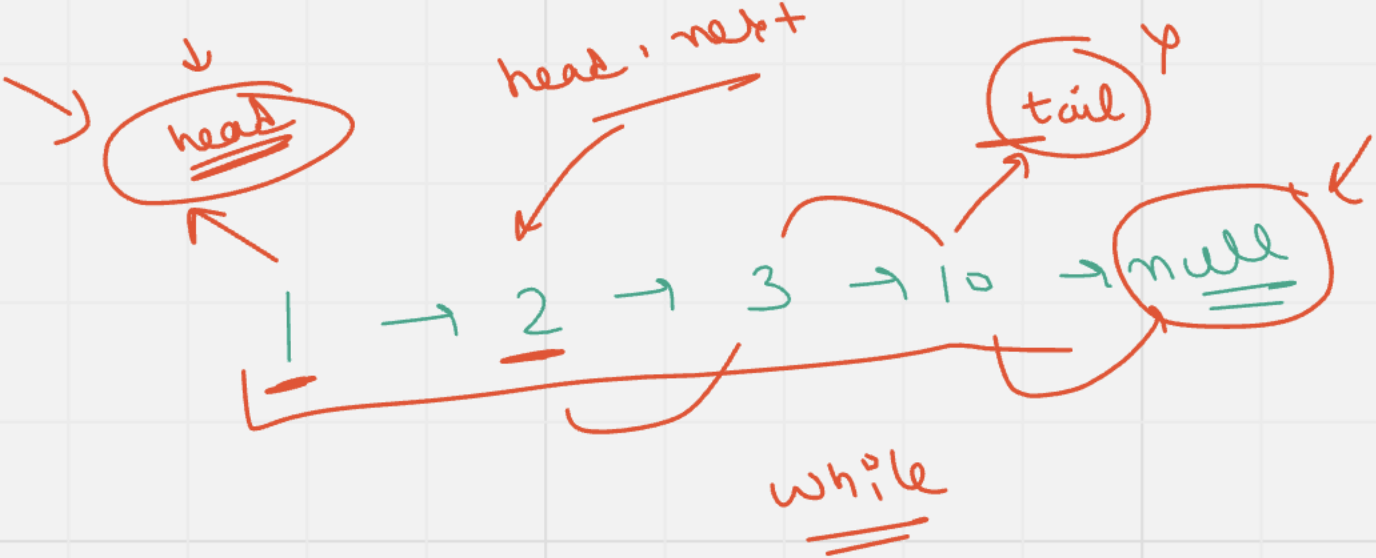
(one.next.next.data) = 3

One.next.next.next.data =

null.data

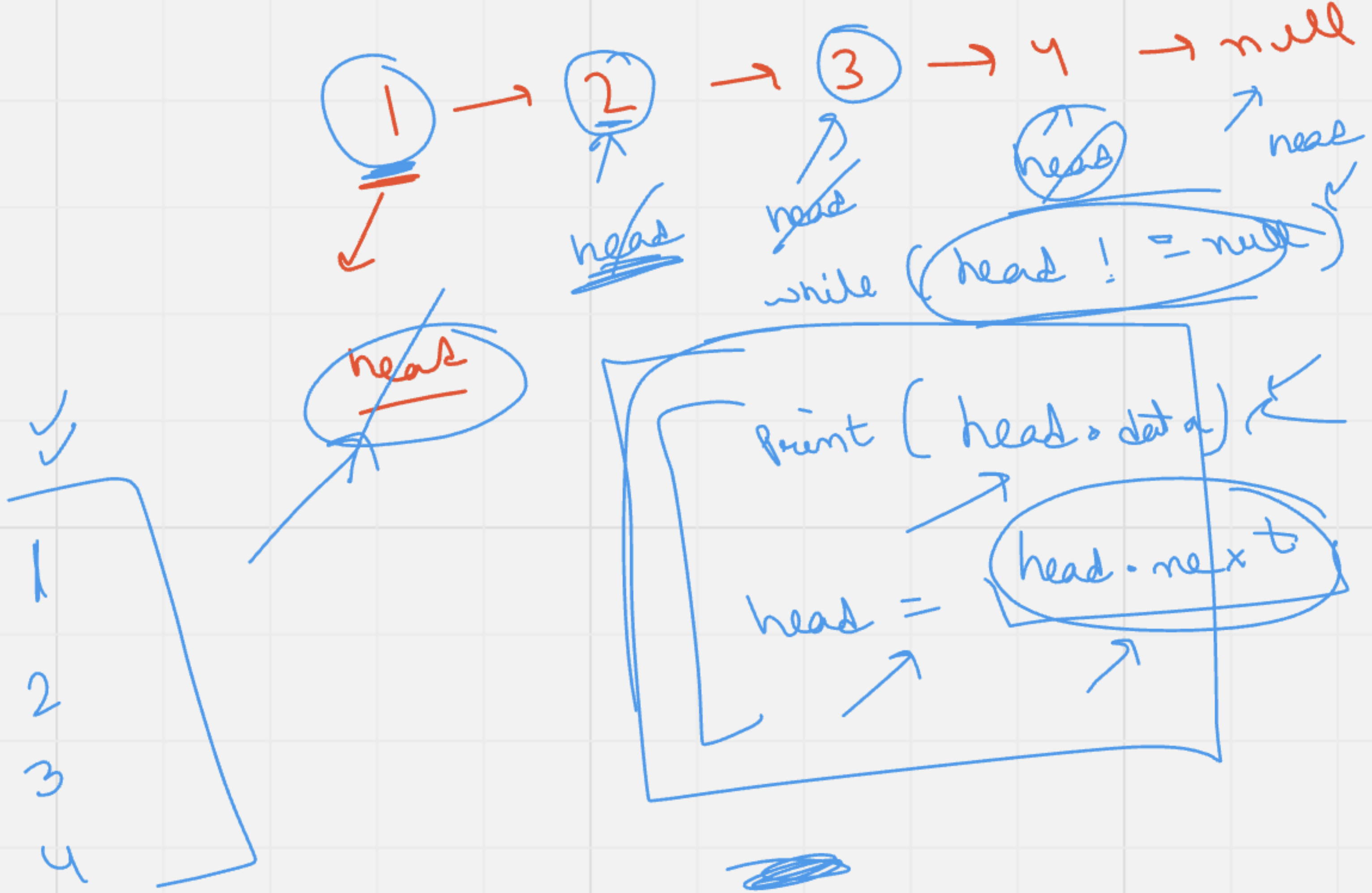
null Pointer exception

next

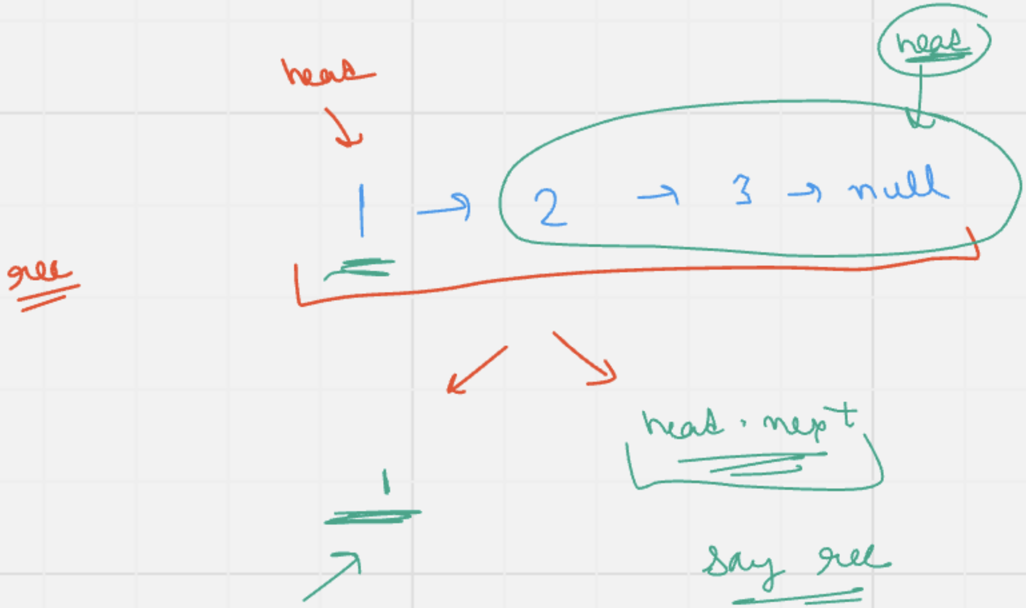


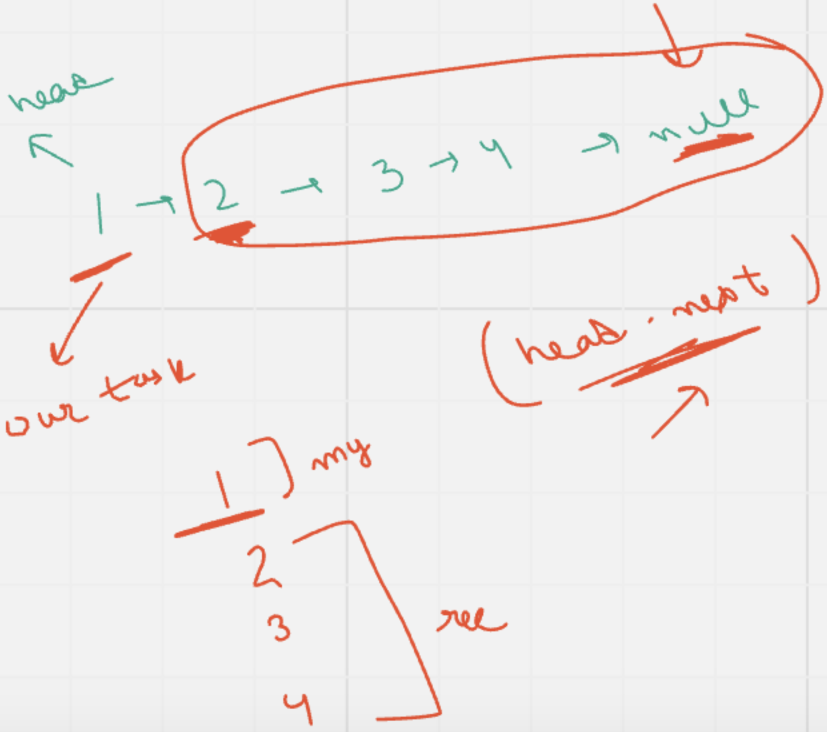
head.next

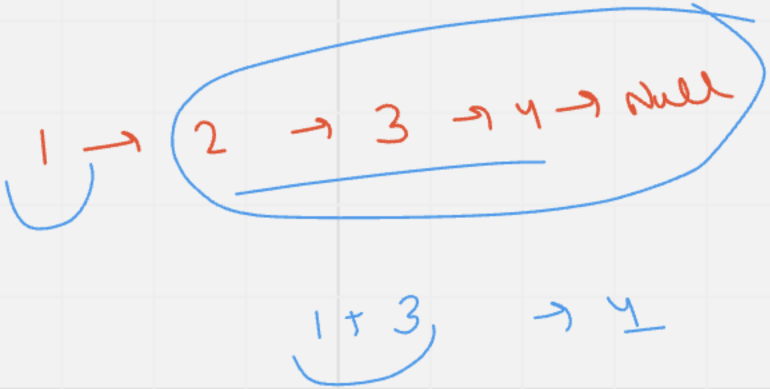
```
static void printLL(Node head) {


}
```

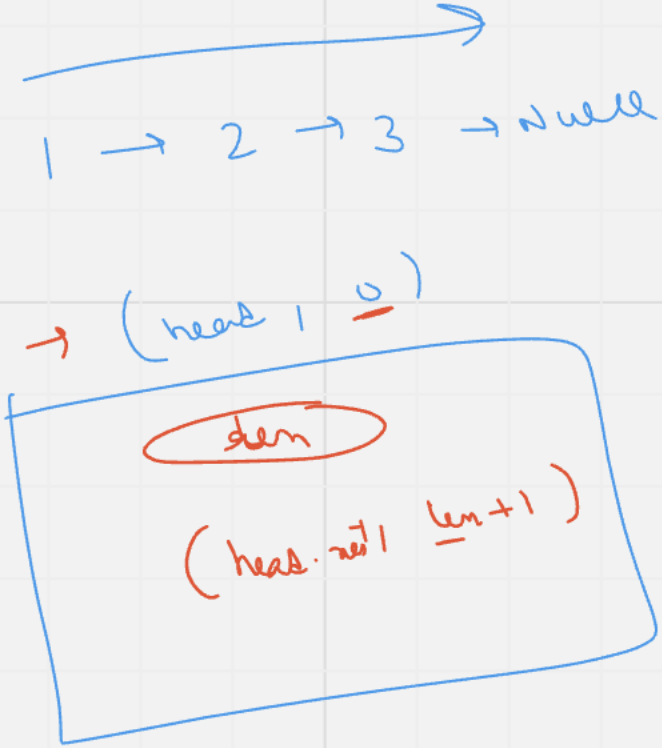


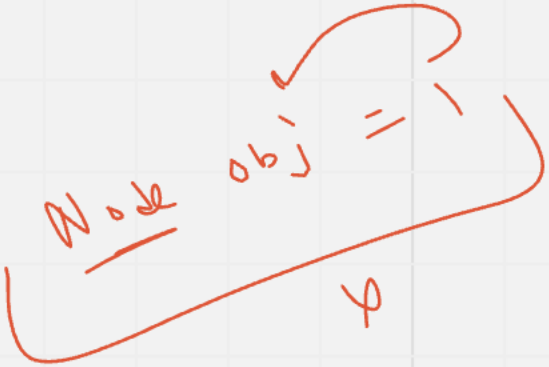
```
while ( head ! = null )
    print ( head . data )
    head = head . next
```

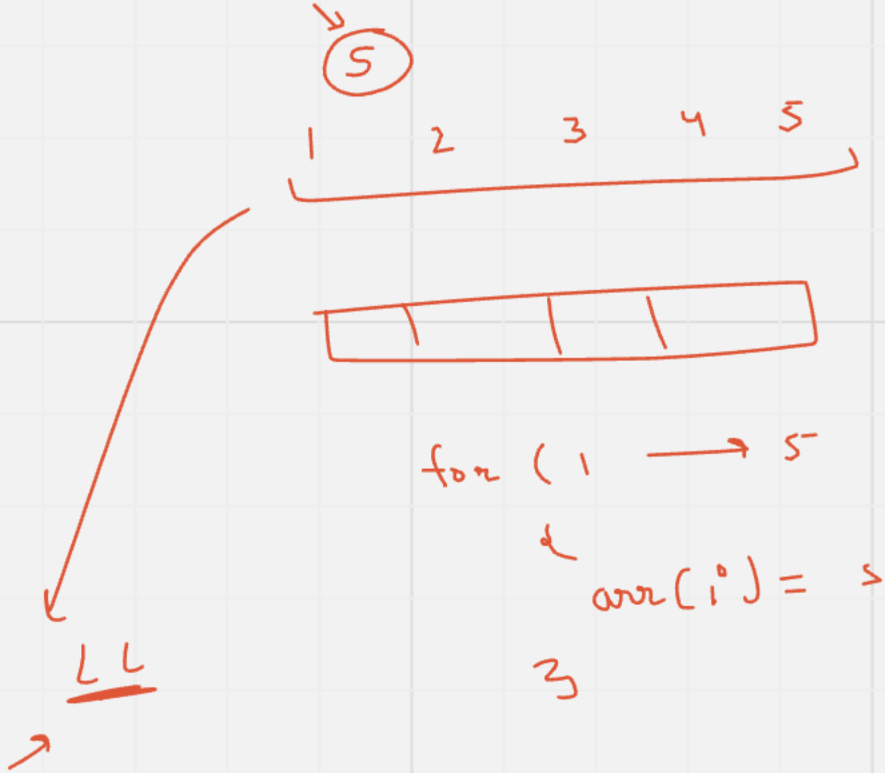

head
head
rec
1 → 2 → 3 → null
1
head.next
say rec


head
1 → 2 → 3 → 4 → null
our task
(head.next)
1 my
2
3
4
rec

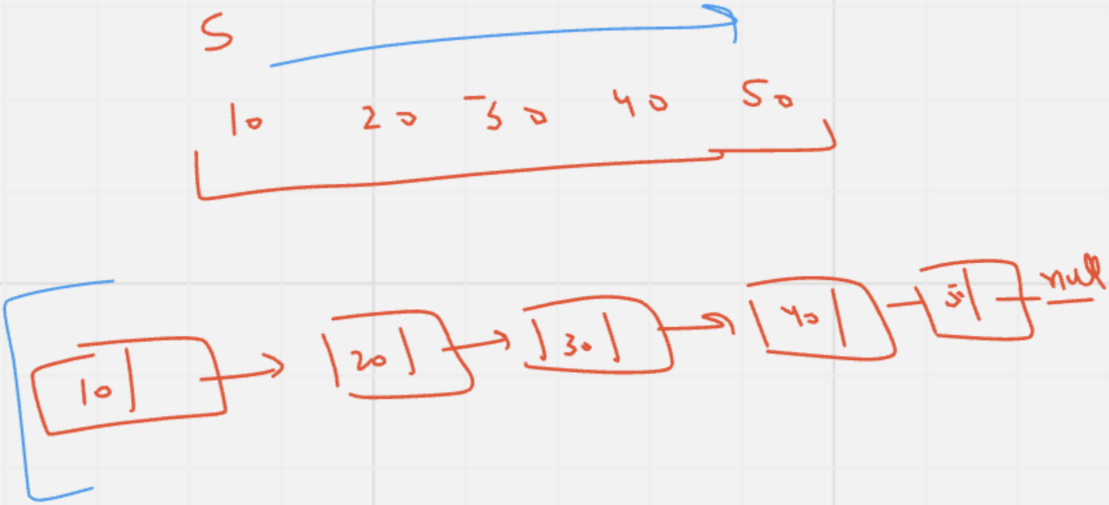

1 → 2 → 3 → 4 → Null
1+3 → 4

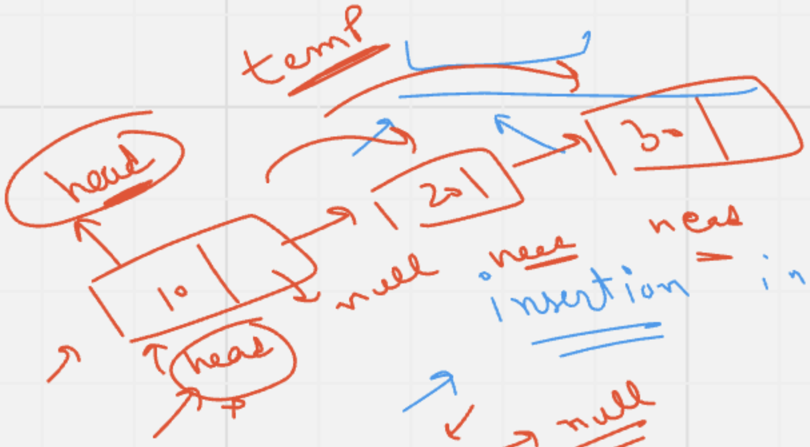

1 → 2 → 3 → Null
→ (head, 0)
len
(head.next, len+1)


Node obj = 1
φ



```
for ( 1 → 5 )
{
  arr(i) = s.nextInt()
}
```





Insertion in LL

Node head → null

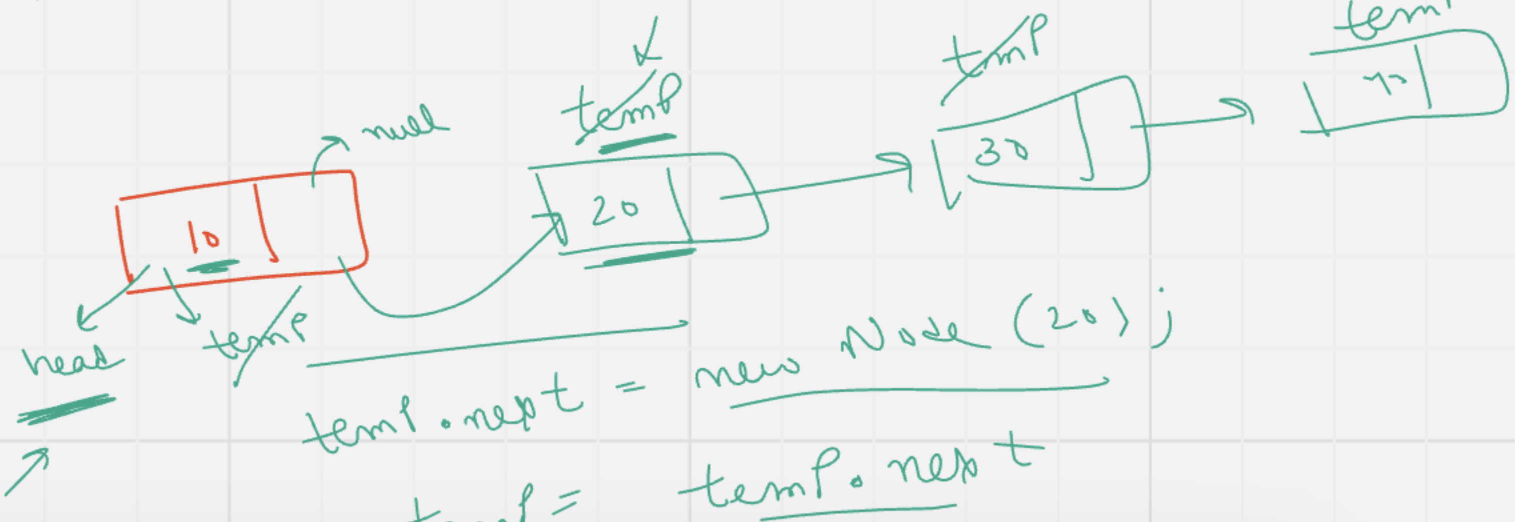
Node tmp → null

empty

contain some

```
if (head == null)
    head = new Node(10);
    ~~tail~~ =
    temp = head

else {
    temp.next = new Node(20);
    tmp = tmp.next
}
```

40



```
temp.next = new Node(20);
temp = temp.next
```