LL

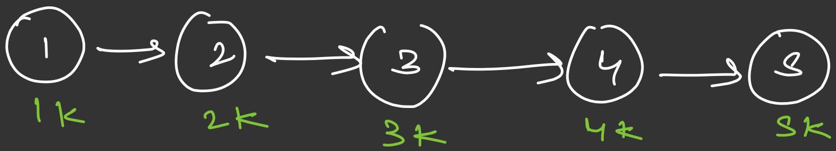
Reverse → Reverse data iterative

Reverse pointer iterative

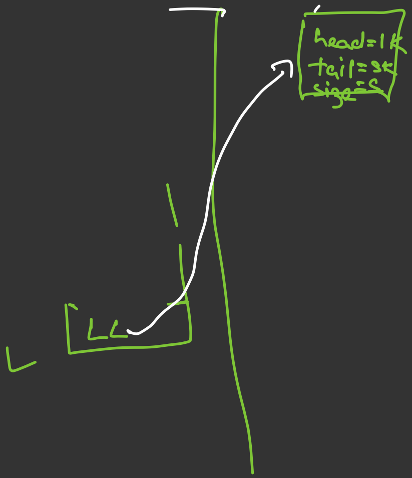
Reverse pointer recursive

→ Reverse data recursive

→ Print Reverse recursively



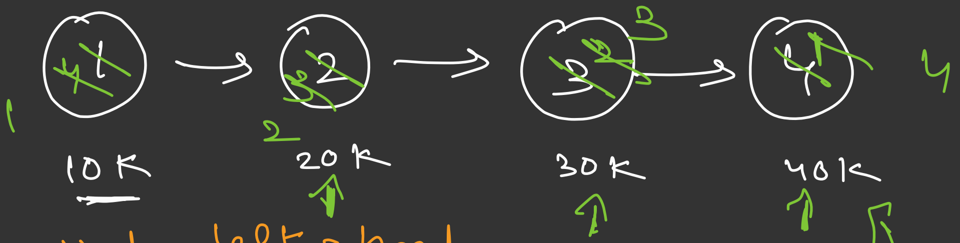
```
6   5   void PR(Node n) {
        if(n==null) return;

     → PR(n.next)
        Syso(n.data); 2.
}
```



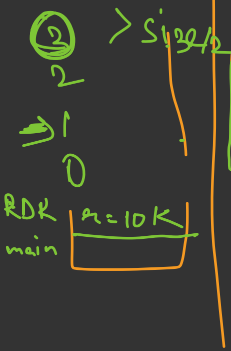
Console

1 2 3 4 5 5 4 3 2 1

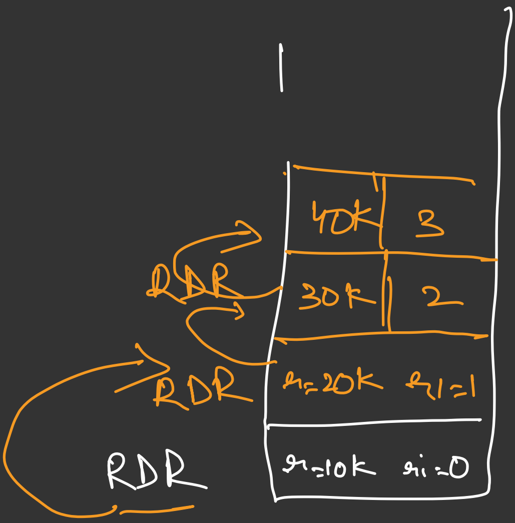
Reverse data recursively



```
b      Node left = head
b        void RDR(Node right, int ri)
  if(right==null) return;
     RDR(right.next, ri+1);
        int t = left.data;
        left.data= right.data;
        right.data =t;
  → left = left.next;
  }
```



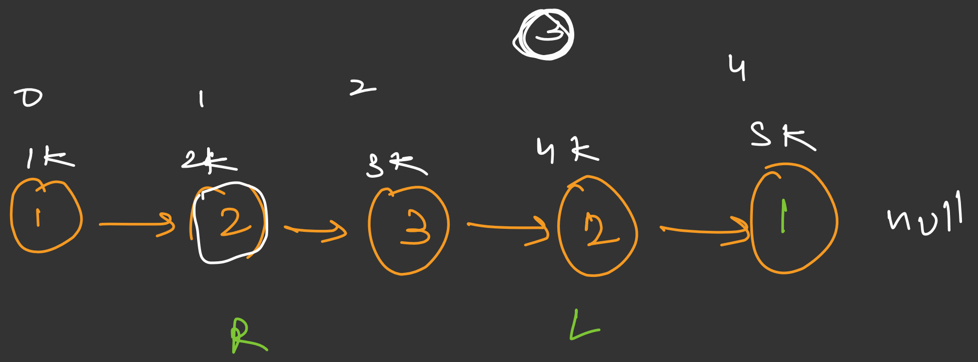
size=4
head = 10K
tail = 40K
left = ~~10K~~
~~20K~~
~~30K~~
40K



Ques Is Palindrome

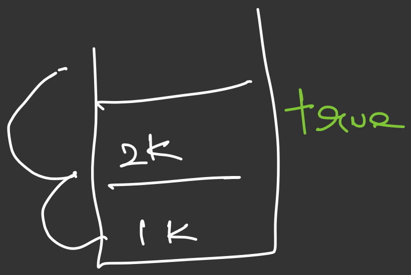
b boolean isPalindrome(Node r, int ri){

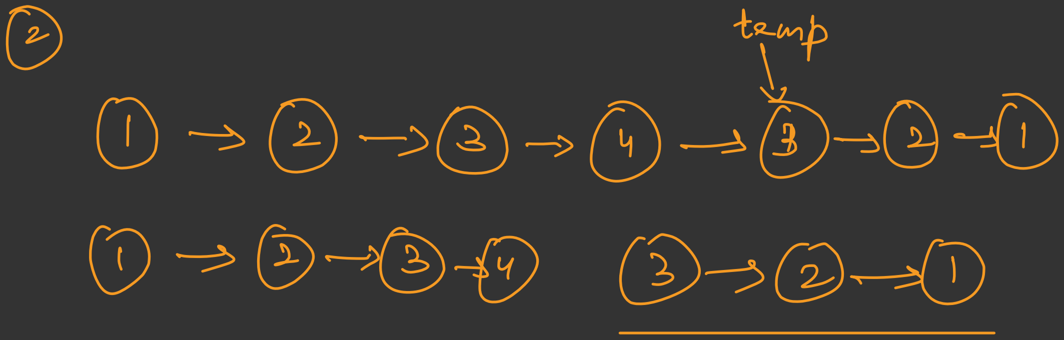
if



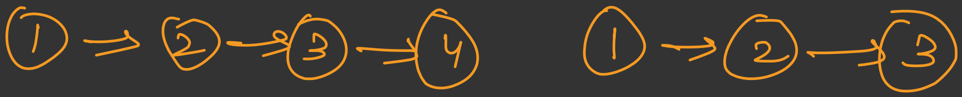
5/2 => <

ri < 2





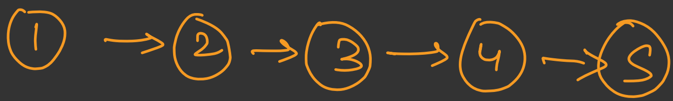
temp = mid.next;

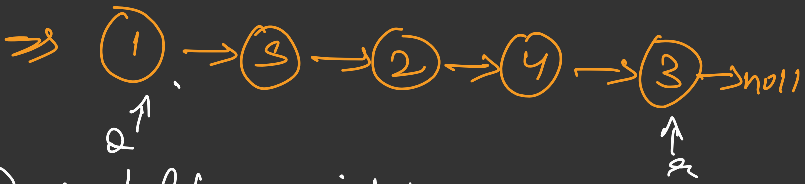


Ques Fold LL

1 - 2 - 3 - 4 - 5 - 6 - 7 - 8

1 - 8 - 2 - 7 - 3 - 6 - 4 - 5

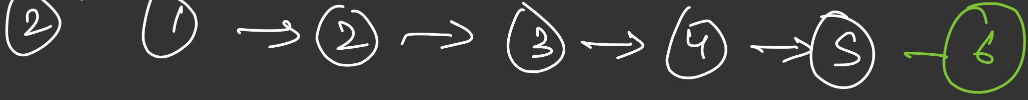


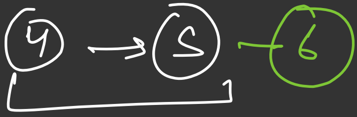


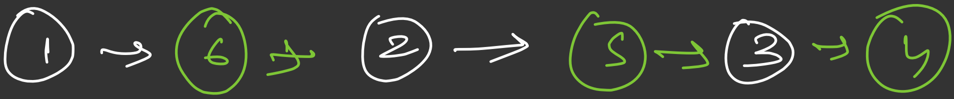
① → left right

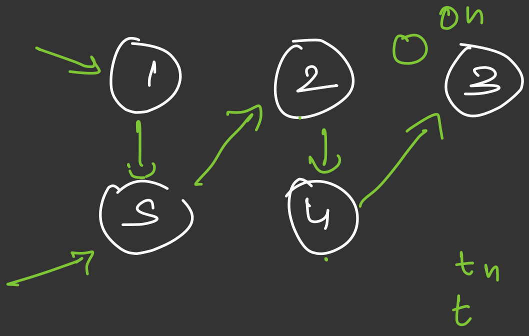
```
Node temp = l.next;
left.next = r;
r.next = temp;

l = l.next;
```

1) Break in 2
2) Reverse 2nd half
3) Merge alternativ





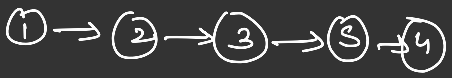




```
On = O.next
tn = t.next;

O.next = t;
t.next = on

O = on
t = tn
```

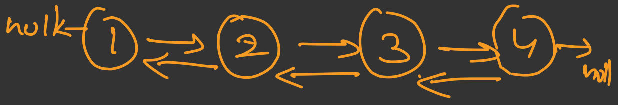
Types of LL

1. Singly LL



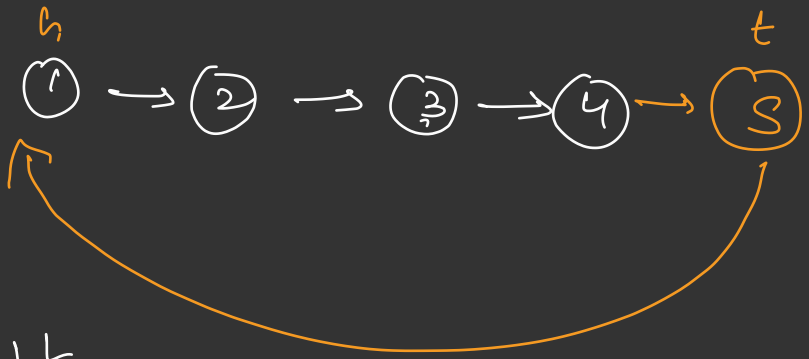
2. Circular LL

3. Doubly LL

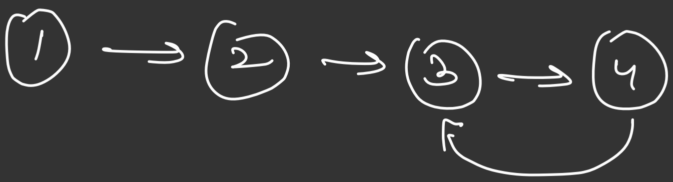


```
node {
  data
  next
  prev
}
```

② Circular LL



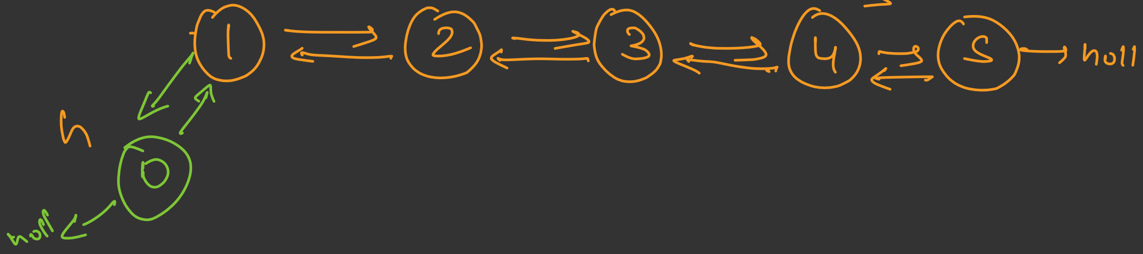
Singly LL



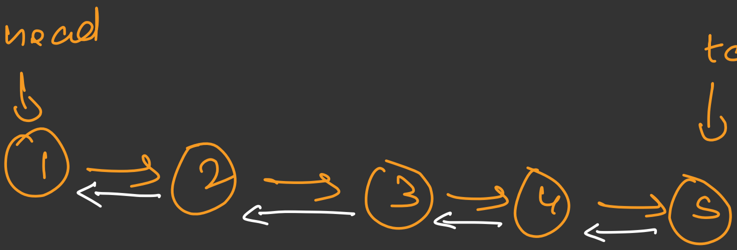
```
n = new Node()
n.next = h
h = n
t.next = h;
```

```
Node {
  int date;
  Node next;
  Node prev;
}
```

③ Doubly LL

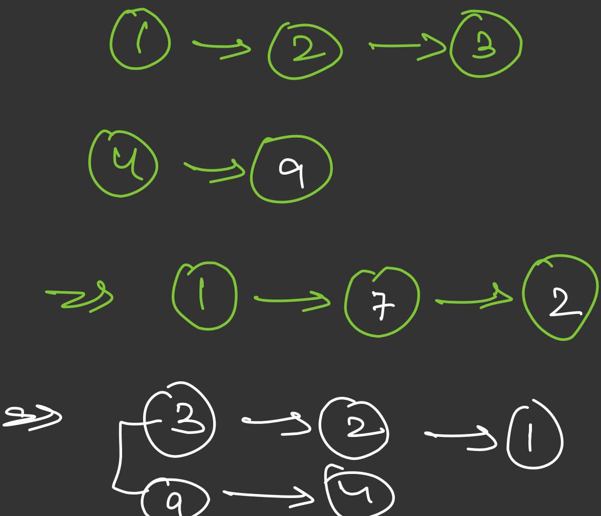


	Singly LL	Doubly LL
add First	O(1)	O(1)
add Last	O(1)	O(1)
add	O(n)	O(n)
remove First	O(1)	O(1)
remove Last	O(n)	$\boxed{O(1)}$
remove	O(n)	O(n)



Ques Add 2 LL

$$\begin{array}{r} 123 \\ 49 \\ \hline 172 \end{array}$$

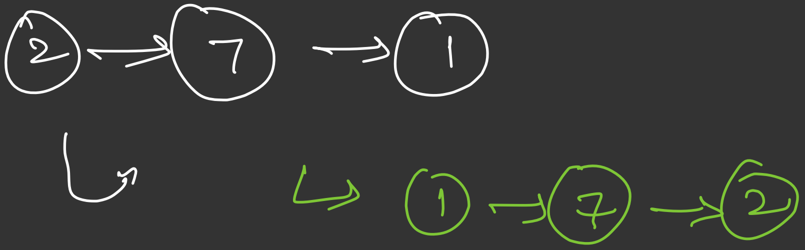


Val = ~~0 1 12~~ 123

0x10 + 1 = 1

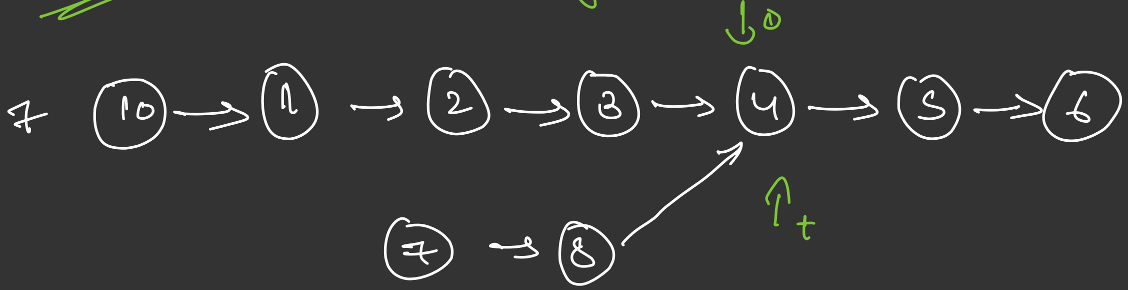
1x10 + 2 = 12

ans = 12

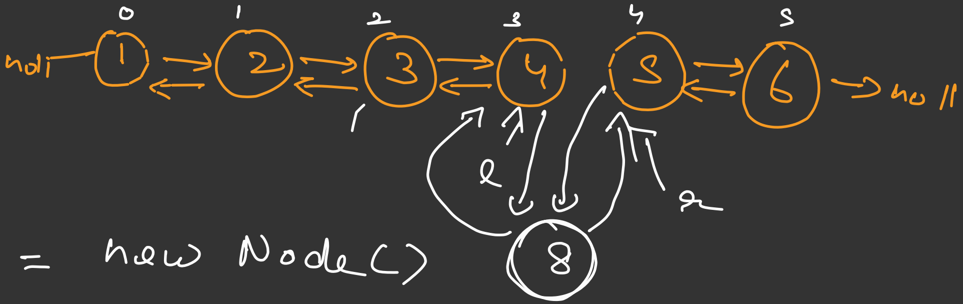


```
c= ans / 10
int c=0;
while(l)
{
    ans = o.d + t.d + c
        Node n = new Node();
    n.data = ans % 10;
    c = ans / 10;
}
```

Ques Intersection of 2 LL



Q add (8, 4) in doubly LL



8

n = new Node()

l = getNode(3);

r = l.next

n.prev = l,

n.next = r;

l.next = n;

r.prev = n;