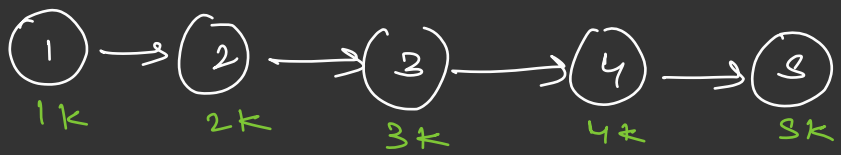
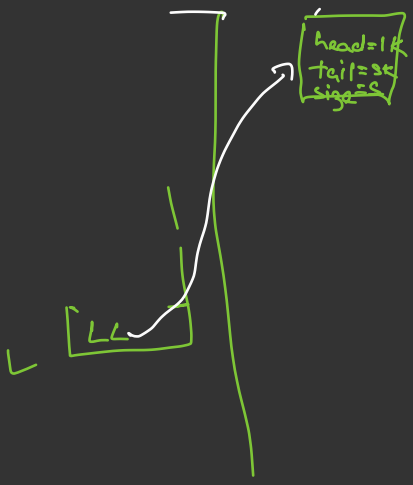
LL

Reverse → Reverse data iterative
Reverse pointer iterative
Reverse pointer recursive
→ Reverse data recursive
→ Print Reverse recursively



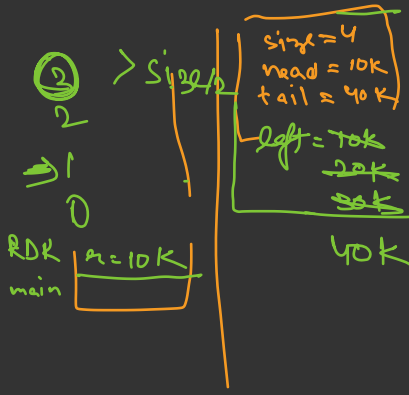

```
void PR(Node n){
  if(n==null) return;
  PR(n.next)
  Sysout(n.data);
}
```

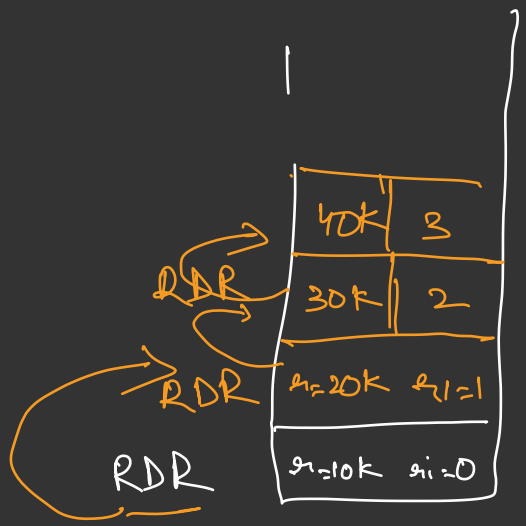


Console
1 2 3 4 5 5 4 3 2 1

Reverse data recursively


```
Node left = head
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;
}
```

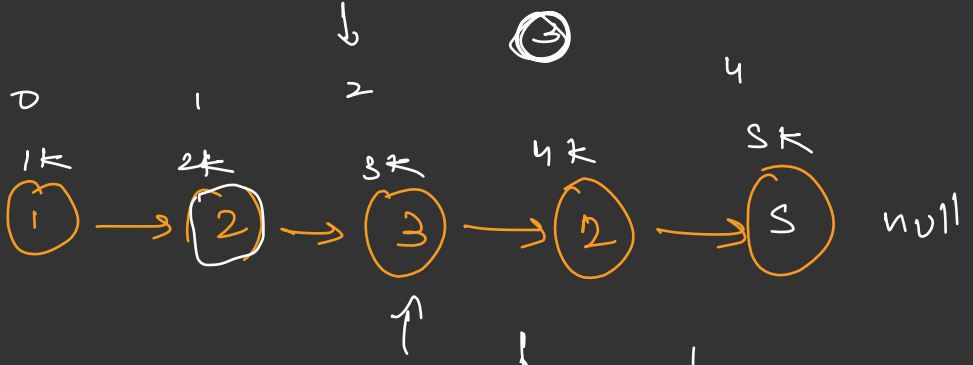


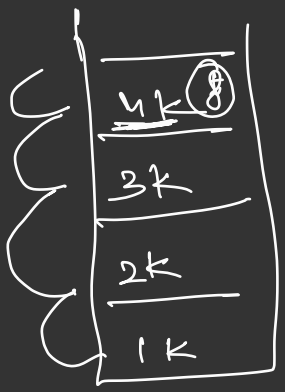


Ques Is Palindrome

```
boolean isPalindrome(Node n, int ni) {
```

if





s/2 >> <

ni < 2

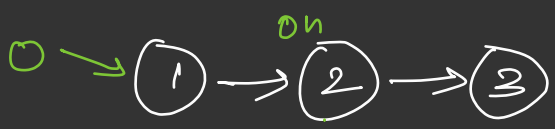
(2)

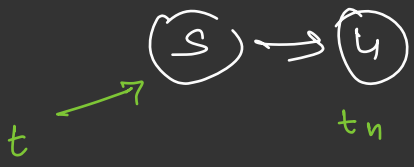
temp = mid.next;


Ques Fold LL

1 → left right

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

l = l.next;
```





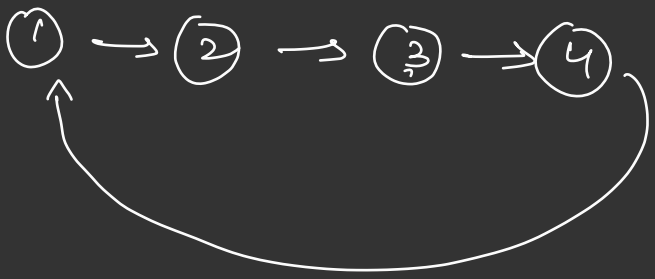

```
on = 0.next
tn = t.next;

0.next = t;
t.next = on

0 = on
t = tn
```

Types of LL

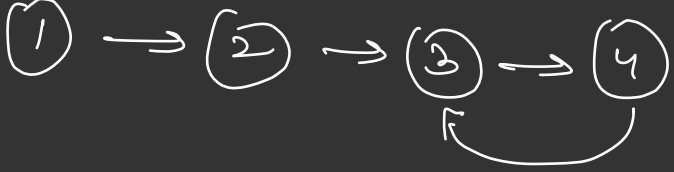
1. Singly LL

2. Circular LL
3. Doubly LL

(2) Circular LL



Singly LL

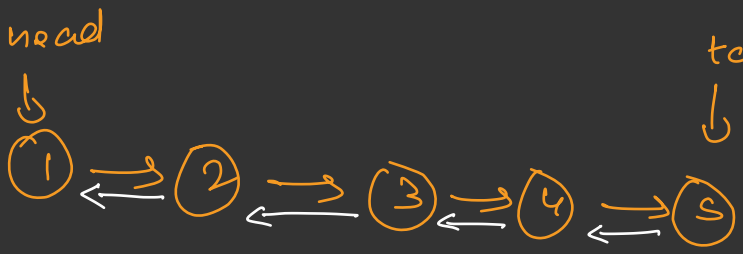



```
n = new Node()
n.next = h
h = n
t.next = h;
```

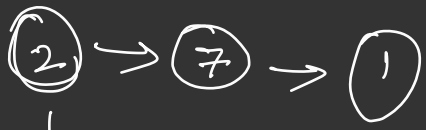
```
Node {
    int data;
    Node next;
    Node prev;
```

(3) Doubly LL

	Singly LL	Doubly LL
addFirst	O(1)	O(1)
addLast	O(1)	O(1)
add	O(n)	O(n)
removeFirst	O(1)	O(1)
removeLast	O(n)	O(1)
remove	O(n)	O(n)



Ques Add 2 LL



ans = 12

c = ans / 10

int c = 0;

while(

ans = o.d + t.d + c

Node n = new Node;

n.data = ans % 10;

c = ans / 10;

Ques Intersection of 2 LL