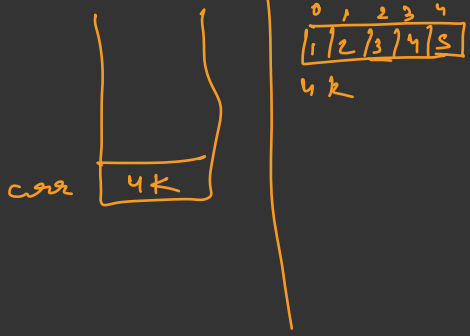
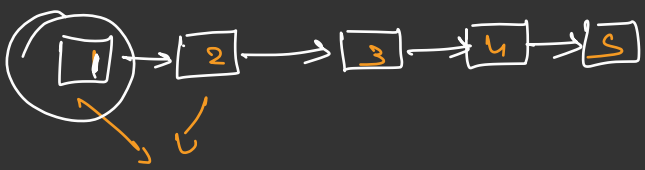
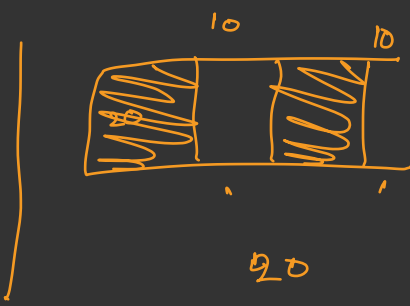
Linked List

int arr = new int[5]

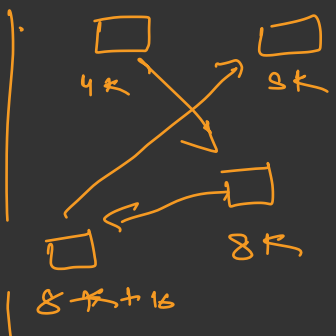


LL was introduced





Node



LL

Node head;

Node tail;

int size

display

add

remove

→ addFirst (int val)

→ addLast (int val)

→ add (int val, int idx)

get

update

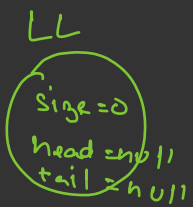
addFirst (1)

(1) Create new node with 1 as data;

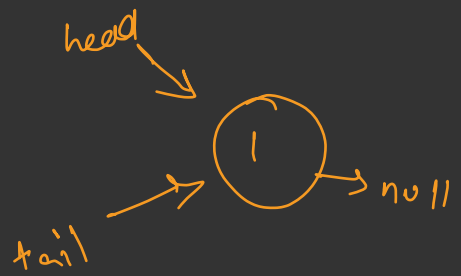
(2) point the next to head

(3) Point head to new node

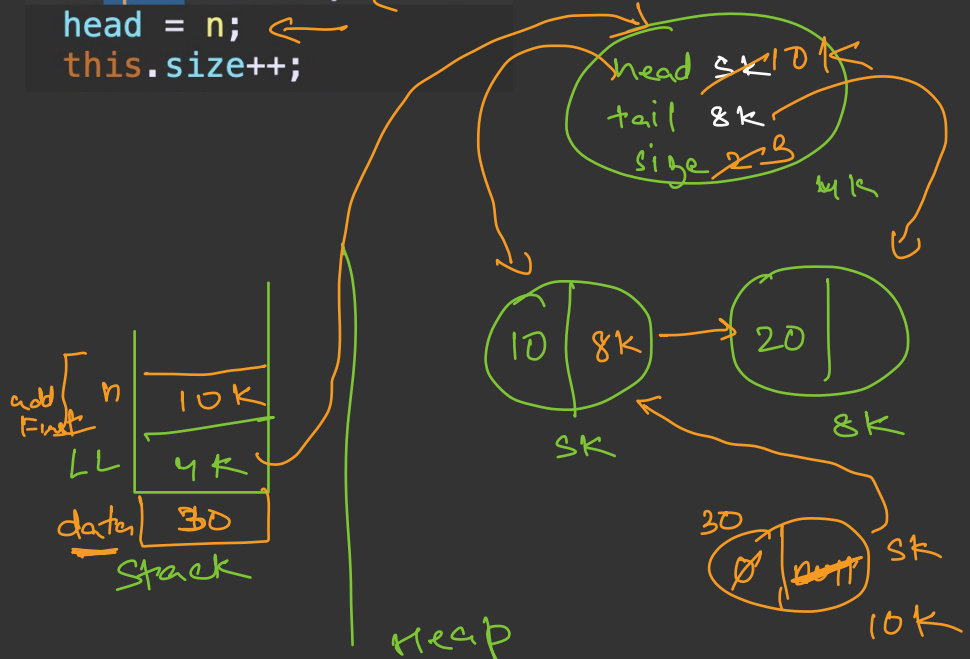
(4) Increase size



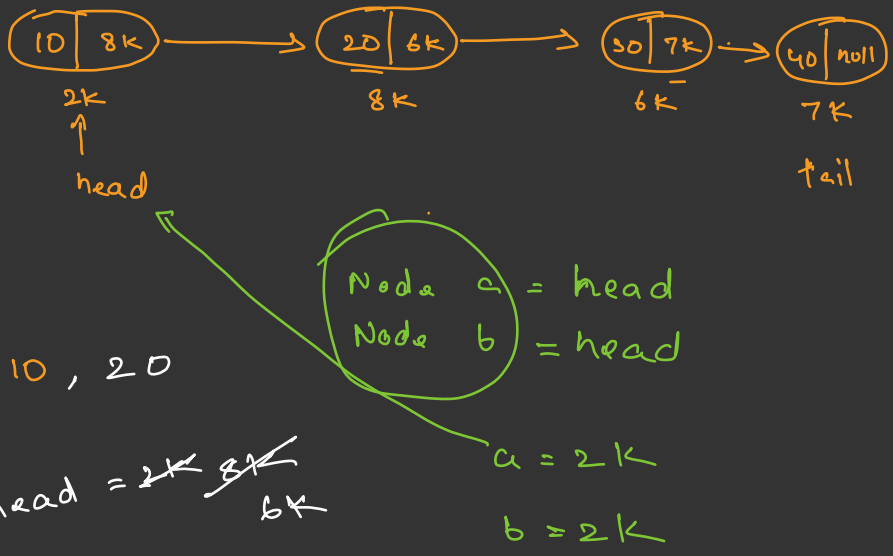

```
if (size == 0)
    Node n = new Node(1, null);
    head = n;
    tail = n
```



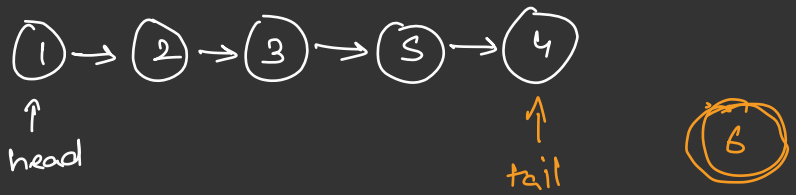

```
Node n = new Node();
n.data = data;
n.next = head;
head = n;
this.size++;
```



```java
public void display() {
    Node temp = head;
    while(temp != tail) {
        System.out.print(temp.data + " -> ");
        temp = temp.next;
    }
    System.out.println("END");
}
```



addLast (int val)



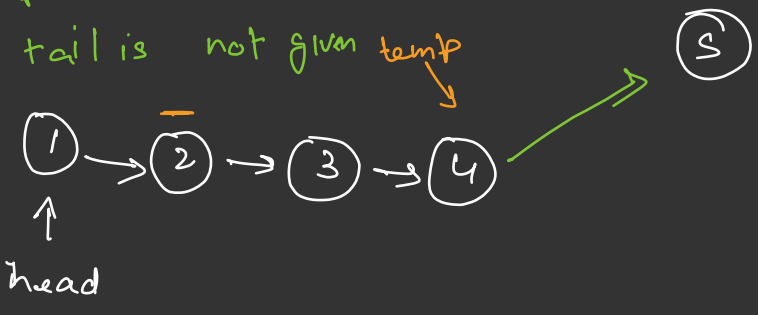
1. Create a node
2. point tail.next to new node

(3) Point tail to new node

(4) Increase size;

Note Special Case

tail is not given temp

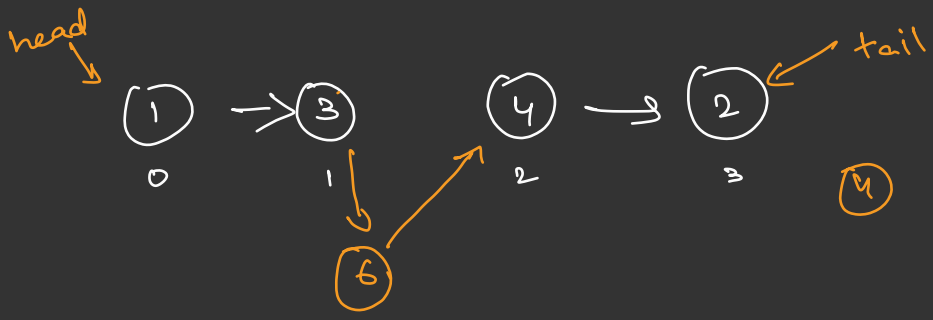



```
Node temp = head;
while (temp.next != null) {
    temp = temp.next;
}
Node n = new Node(data, null);
temp.next = n;
return n
```

add(6, 2)

add (int data, int idx)

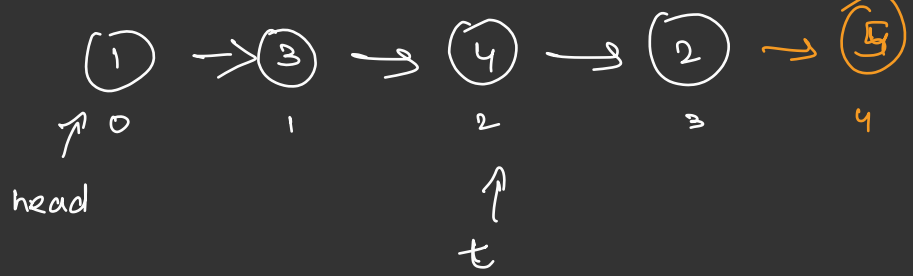
Output →



1. Traverse LL till idx - 1;
2. Create new node and point t.next
3. point t to new node
4. Increase size

add (6, 3)

idx = 3.



i = 0 1 2

```
while (i < idx - 1) {
    t = t.next;
    i++;
}
```

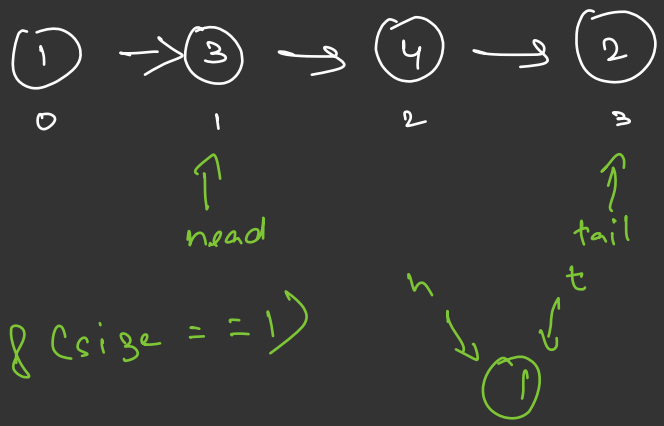
Get(int idx)

1) Traverse till idx
2) return node.data

getNode

1
2 return node.

Remove First



if(size == 1)

s = 0

head = null
tail = null