

---

---

---

---

---



# # Linked List

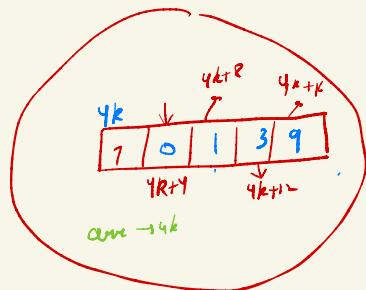
$\text{arr}(0) \rightarrow 4k$   $\text{arr} \rightarrow 7k$

$\text{int } () \quad \underline{\text{arr}_c = \text{new int } [s]}$

$\text{arr} \rightarrow 4k$   $5 \times 4$

$\text{arr}(0) \rightarrow$   
 $\text{arr}(1) \rightarrow 4k+4$

$4k \rightarrow 4k+4 \rightarrow$



data  
address of next  
Node object

Student s



class Node {

    → int data;

    → Node next;

}

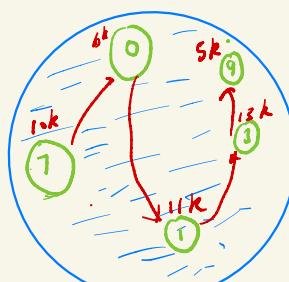
{ Node f = new Node();

f.data = 7

2) Node s = new Node();

s.data = 0

$s = 6k$

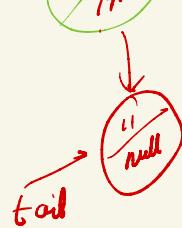


$s = 6k$



$\text{head} = 10k$

```
public void addLast(int data){
    1) Node nn = new Node(data);
    2) tail.next = nn;
    3) tail = nn;
```



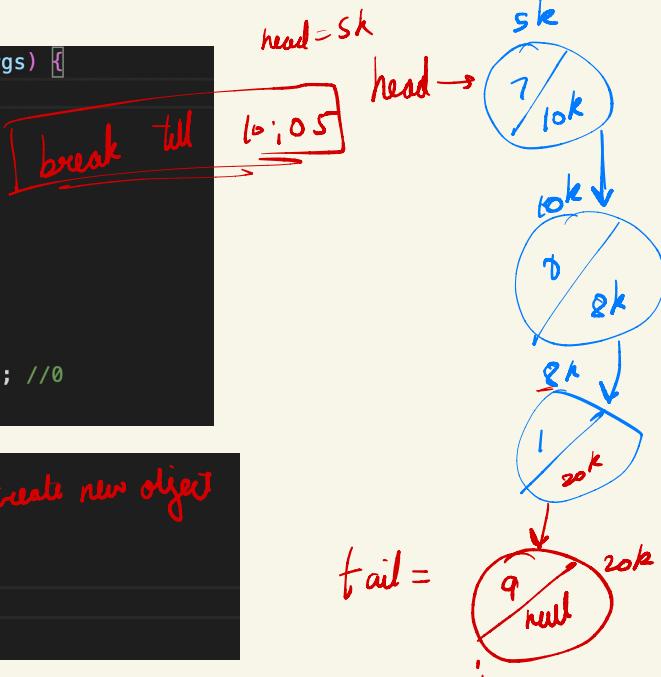
$\text{tail}$

```

public static void main(String[] args) {
    1) Node f = new Node(d: 7);
    2) Node s = new Node(d: 0);
    3) Node t = new Node(d: 1);

    // connecting first and second
    4) f.next = s;      head=f
    5) s.next = t;      tail=t
    6) System.out.println(f.next.data); //0
}

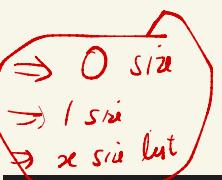
```



```

public void addLast(int data){ → q
    1) Node nn = new Node(data); → create new object
    tail.next = nn;
    tail = nn;
}

```



```

public class Main {
    1 static Node head;
    2 static Node tail;

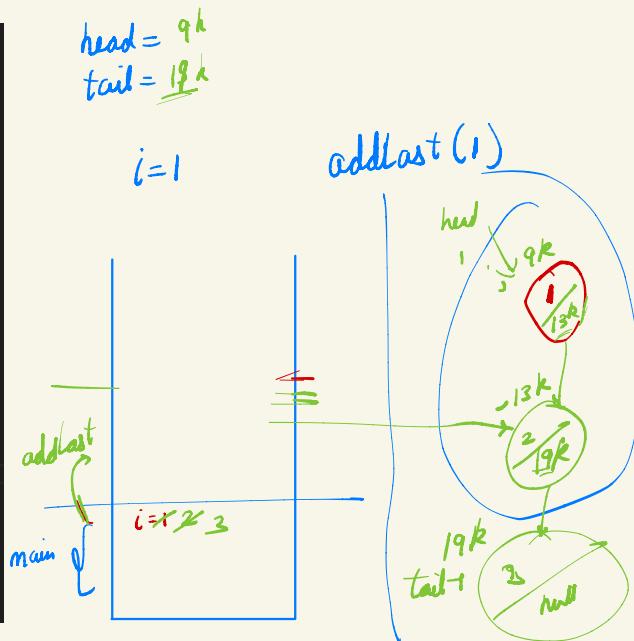
    public static void addLast(int data){
        1) Node nn = new Node(data);

        2) if(head == null){ // in case of 0 size ll
            head = nn; // this is my first node of list
            tail = nn; // and also my last node of list
        } else {
            → tail.next = nn; // attaching
            → tail = nn; // shifting tail to last
        }
    }

    Run | Debug
    public static void main(String[] args) {
        1) head =null;
        tail = null;
        2) for(int i=1; i<=3; i++){
            addLast(i);
        }

        System.out.println(head.data);
        System.out.println(tail.data);
    }
}

```



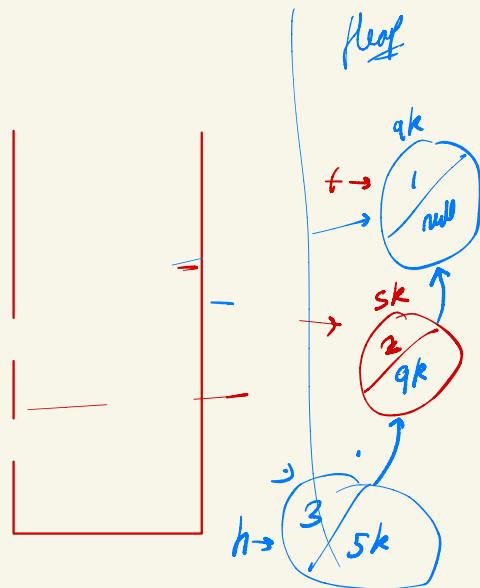
Ques Add a node at the start of ll.

- 1) addFirst(1);
- 2) addFirst(2);
- 3) addFirst(3);

head = 5k  
tail = 9k

```
public static void addFirst(int data){
    Node nn = new Node(data);

    if(head == null){
        head = nn;
        tail = nn;
    } else {
        nn.next = head;
        head = nn;
    }
}
```



# display



```
// homework |
public static void display(){
    Node curr = head;

    while(curr!=null){
        System.out.print(curr.data+" ");
        curr=curr.next;
    }
}
```

1 3 5 7 4

curr = 1/qk & tail = 4/qk null

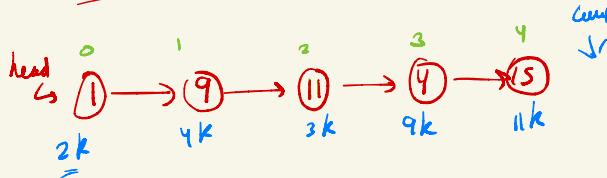
Ques Remove first element from linked list.



head = null

$\Rightarrow \underline{\underline{\text{head} = \text{head}.next;}}$

Obe Get Node at particular index.



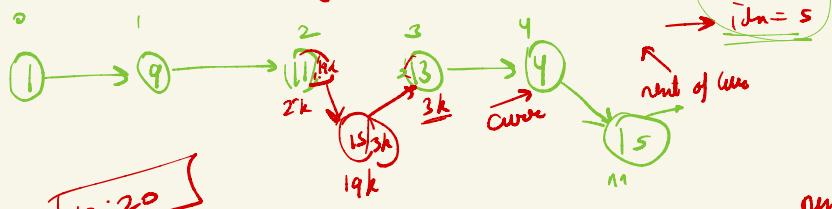
- 1)  $idx = -1$   $\rightarrow idx = 6$   
 2)  $idx = 0$   
 3)  $idx = 2$   
 4)  $idx = 4$

```
public static Node getAt(int idx){  
    if(idx < 0){  
        System.out.println("given index is out of range");  
        return null;  
    }  
  
    Node curr = head;  
    for(int i=0; i<idx; i++){  
        if(curr==null){  
            System.out.println("given index is out of range");  
            return null;  
        }  
        curr=curr.next;  
    }  
  
    return curr;  
}
```

curr = head

- $i = 0 \quad \underline{i < 6} \quad \checkmark$   
 $i = 1 \quad \checkmark$   
 $i = 2 \quad \checkmark$   
 $i = 3 \quad \checkmark$   
 $i = 4 \quad \checkmark$   
 $i = 5 \quad \checkmark$

Obe <sup>use</sup> add a Node at given index.



next of curr = null

```
public static void addAt(int idx, int data){  
    Node nn = new Node(data);  
  
    Node curr = getAt(idx-1);  
    Node nextOfCurr = curr.next;  
  
    curr.next= nn;  
    nn.next = nextOfCurr;  
}
```

$i \leq idx-1$

idx = 0

$2k.next = 19$   
 $NOC = 3k$

$nn.next = nextOfcurr$

$\sim 1$

$19k.next$

of ll.

Obe Remove and return the last Node of ll.

curr = 2k 3k 4k

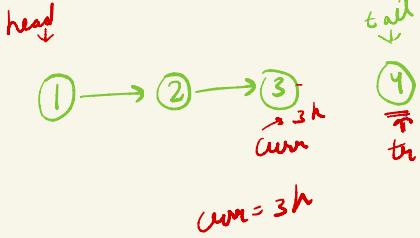
```

public static Node removeLast(){
    if(head==null){ // 0 size
        return null;
    } else if(head==tail){ // 1 size
        Node tr=tail;
        removeFirst();
        return tr;
    } else {
        Node curr = head;
        while(curr.next!=tail){
            curr = curr.next;
        }

        curr.next=null;
        // updating tail
        Node tr = tail;
        tail = curr;

        return tr;
    }
}

```

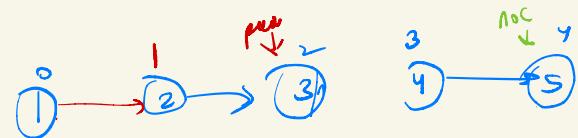


4 is deleted from ll  
but not from heap

wrote on your own code ↴ test

dry run the code

Ques Remove at given index.



```

public static void removeAt(int idx){
    int size = getLength();
    if(idx==0){
        removeFirst();
    } else if(idx==size-1){
        removeLast();
    } else {
        Node prev = getAt(idx-1);
        prev.next = prev.next.next;

        // Node nextOfCurr = prev.next.next;
        // prev.next = null;
        // prev.next = nextOfCurr;
    }
}

```

```

public static int getLength(){
    Node curr = head;
    int len = 0;

    while(curr!=null){
        curr = curr.next;
        len++;
    }

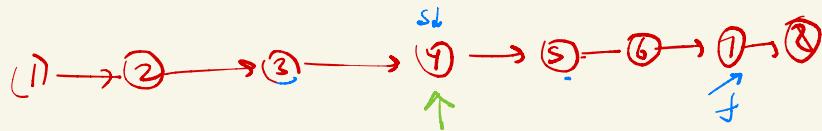
    return len;
}

```

- 1) These were the building blocks.
- 2) We have a default linked list.

Ques Find middle of linked list.

breadth first  
to

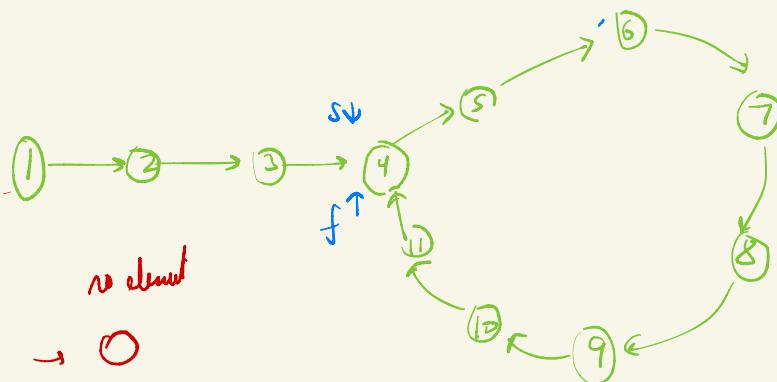


Ques what if we wanted to return first middle

slow → 1 node per move  
fast → 2 nodes per move

[hare and tortoise method]

```
public ListNode middleNode(ListNode head) {
    ListNode slow = head;
    ListNode fast = head;
    while(fast!=null && fast.next!=null){
        1) slow = slow.next;
        2) fast = fast.next.next;
    }
    return slow;
}
```



- 1) Check if there is a cycle
- 2) Keep fast at the intersecting node and take slow back to head
- 3) Move slow and fast at equal pace.
- 4) They will coincide again → this is the starting point.

