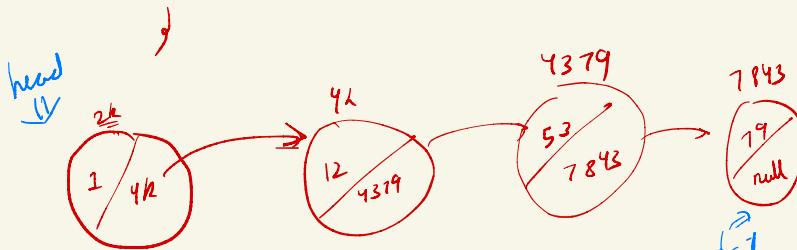
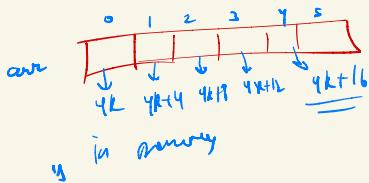



Linked List

```
class Node {
    int data;
    Node next;
}
```



```
public static void main(String[] args) {
    Node head=new Node(data: 5);
    Node n1=new Node(data: 45);

    head.next=n1;

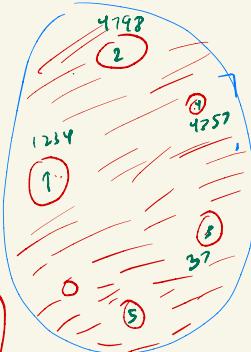
    Node n2=new Node(data: 7);
    n1.next=n2;

    Node nextOfHead=head.next;
    System.out.println(n1+" fggfg "+nextOfHead);
    System.out.println(nextOfHead.data);
}
```

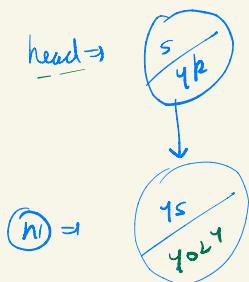
```
class Node {
    int data;
    Node next;

    public Node(){
    }

    public Node(int data){
        this.data=data;
    }
}
```



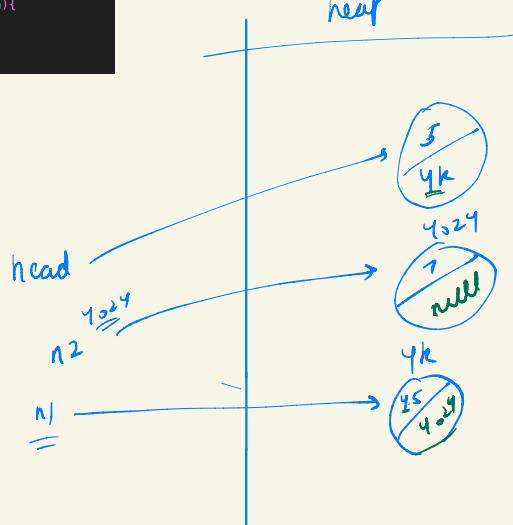
head
head.data
head.next=4k

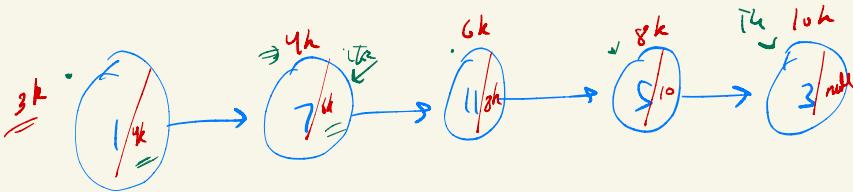


n1.data = 45
n1.next = 4024

next of Head = 4k
n1

n1

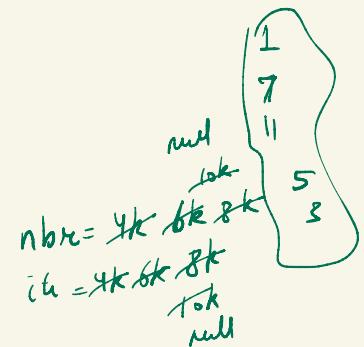




head
(3k)

```
public static void printLinkedList(Node head){
    System.out.println("Printing linked list");
    Node itr=head;

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



$$3 \Rightarrow \underline{1, 2, 3}$$

$$\text{data} = 2, 3$$

itr =

$$f-l = 1$$

```
Scanner scn=new Scanner(System.in);
System.out.println("Enter number of inputs ");
int n=scn.nextInt();

int first_ele=scn.nextInt();

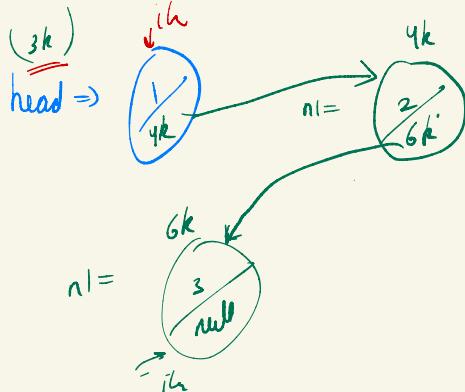
Node head=new Node(first_ele);
Node itr=head;

for(int i=1; i<n; i++){
    int data=scn.nextInt();

    Node n1=new Node(data);
    itr.next=n1;

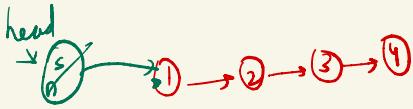
    Node nbr=itr.next;
    itr=nbr;
}

printLinkedList(head);
}
```



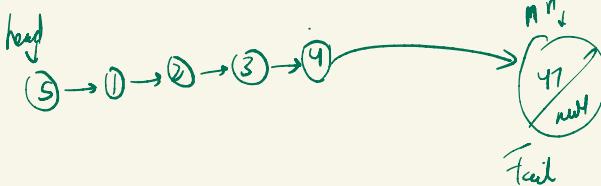
addFirst

val = 5



Node nn=new Node (val);

nn.next=head;
head=nn;

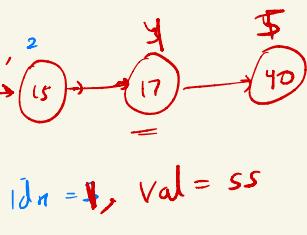
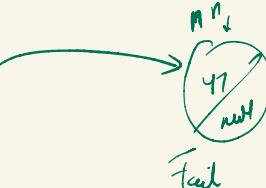


```

public static Node getTail(Node head){
    Node itr=head;
    while(itr.next!=null){
        // itr=itr.next;
        Node nbr=itr.next;
        itr=nbr;
    }
    return itr;
}

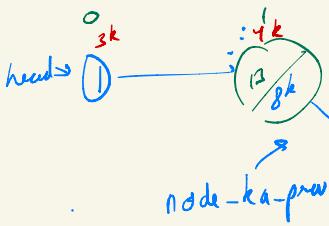
public static void addLast(int val, Node head){
    Node tail=getTail(head);
    Node nn=new Node(val);
    tail.next=nn;
    tail=nn;
}

```



i > n - 1, val = 55

Ans Add a val at idx



```

public static Node getAt(Node head, int idx){
    public static Node removeAt(Node head, int idx){
        if(idx==0){
            return removeFirst(head);
        }

        Node node_ka_prev=head;

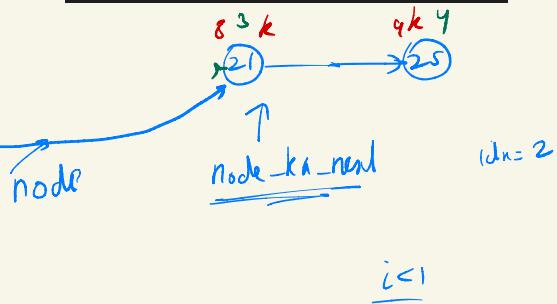
        for(int i=0; i<(idx-1); i++){
            Node nbr=node_ka_prev.next;
            node_ka_prev=nbr;
        }

        Node node=node_ka_prev.next;
        Node node_ka_next=node.next;

        node_ka_prev.next=node_ka_next;
    }

    return head;
}

```

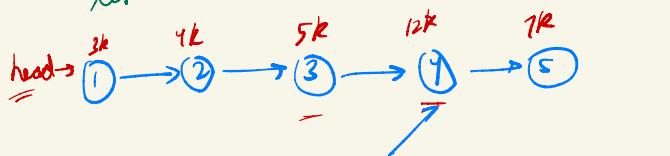


idx = 2

i < 1

Ques1 Find length of linked list.

Ques2 Write a recursive function to return a node in linked list with data equal to target - ①



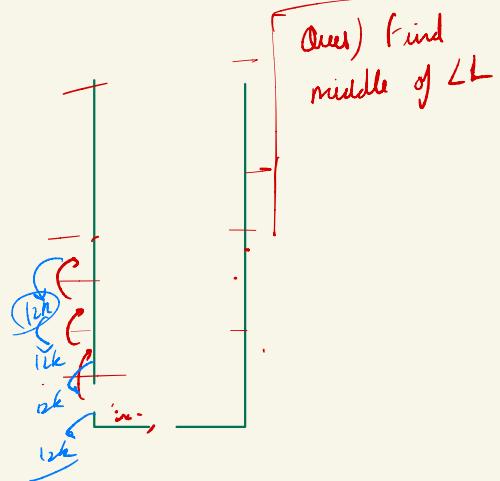
expectation
It will find the node from head to last

$$tar = 4$$

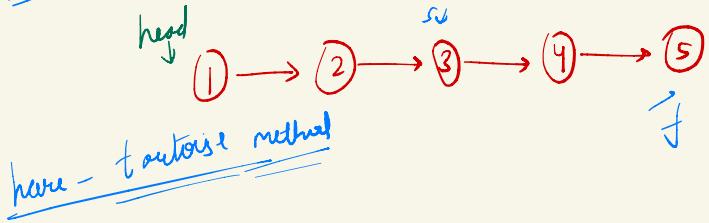
Faith

It will find node from head.next to last.

```
public static Node find_Rec(Node itr, int tar){  
    if(itr==null){  
        return null;  
    }  
  
    if(itr.data==tar){  
        return itr;  
    }  
  
    Node ans=find_Rec(itr.next, tar);  
    return ans;  
}
```



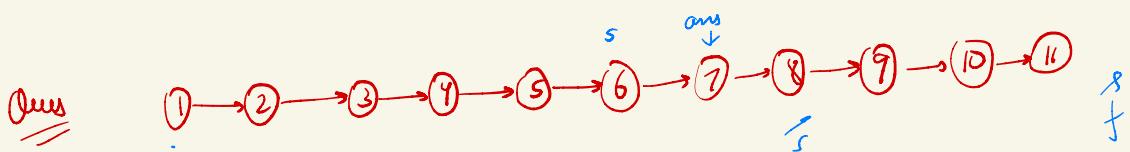
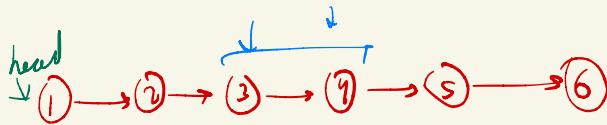
Ques Find middle of LL in one traversal .



```
public static Node getMid(Node head){
    Node slow=head;
    Node fast=head;

    while(fast.next!=null && fast.next.next!=null){
        slow=slow.next;
        fast=fast.next.next;
    }

    return slow;
}
```



```
public static Node getKthFromEnd(Node head, int k){
    Node slow=head;
    Node fast=head;

    1) for(int i=0; i<k; i++){
        2)     fast=fast.next;
    }

    while(fast!=null){
        slow=slow.next;
        fast=fast.next;
    }

    return slow;
}
```

Run | Debug

last = 5th
fast = 9th