Linked List

Recap

OOPS → User defined data type

```
public static class Node{
    int data;
    double percent;
    String name;
}
```

Arrays → Arraylist

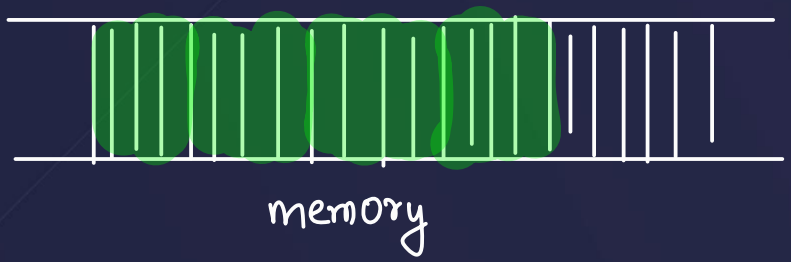
int → 4 bytes → 32 bits

```
int[] arr = new int[4];
```

Limitations

1) Fixed Size → Arraylist

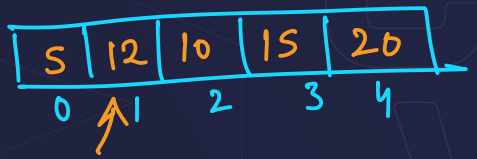
2) Continuous Memory



get → O(1)

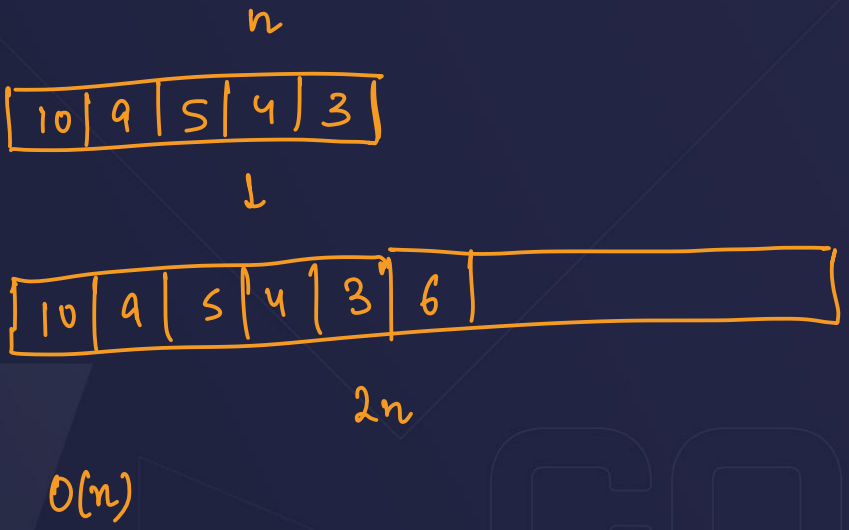
→ for Ex → 100 , arr[57]

3) Insert in between O(n)



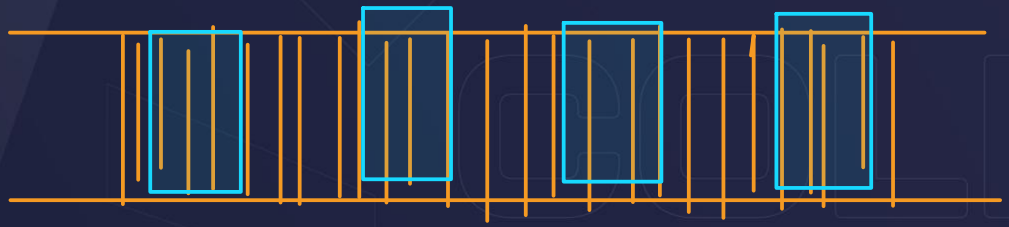
Arrays

Limitations



Arrays

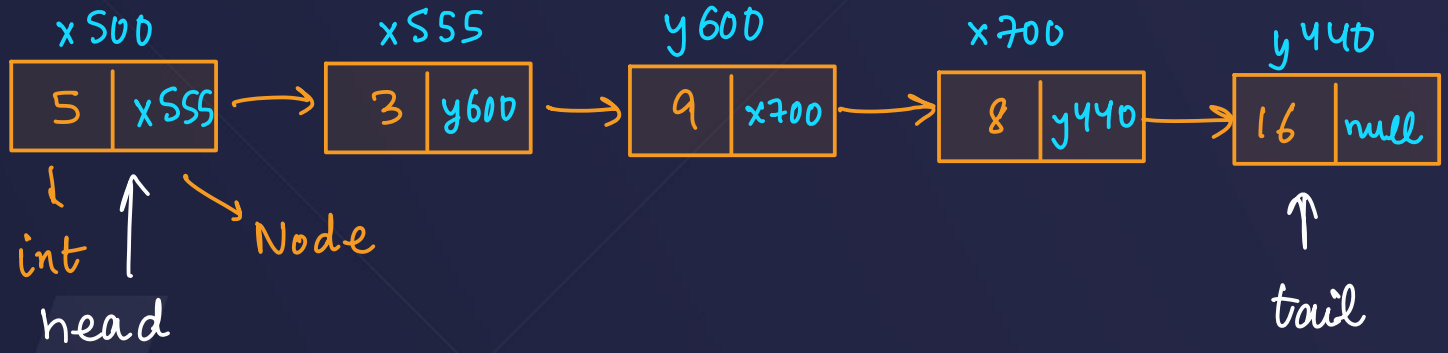
Need for a new linear data structure



Introduction to Linked List

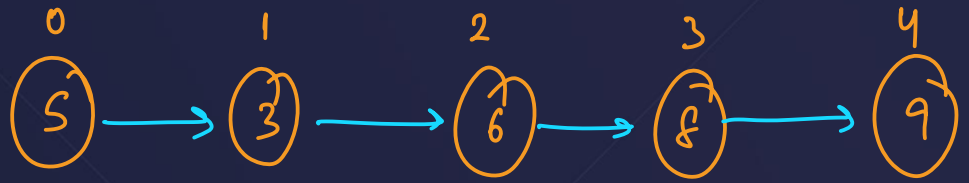
Node

5, 3, 9, 8, 16



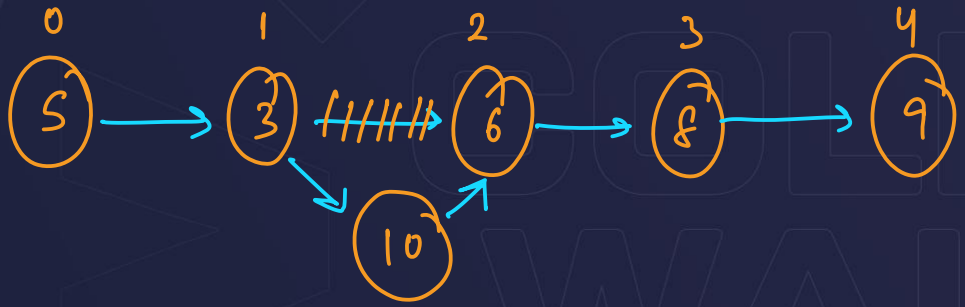
Introduction to Linked List

Does linked list overcomes the limitations of arrays?



get → O(n)

Insertion → 2nd index → 10



List Node

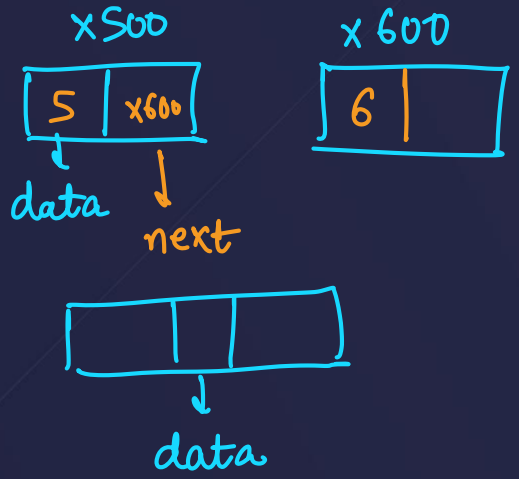
```
public static class Node {
    int data;
    Node next;

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

Types of Linked List

1) Singly Linked List

2) Doubly Linked list

3) Circular Linked List

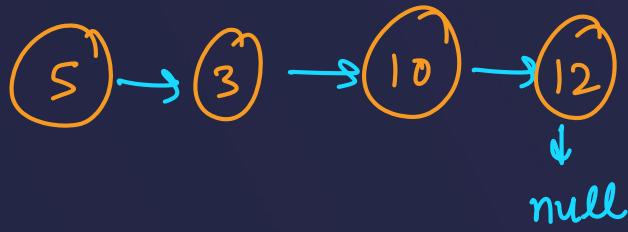


Singly linked list

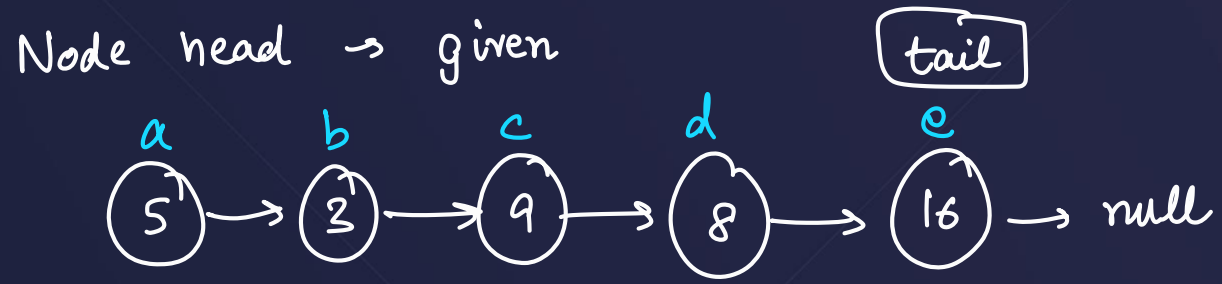
Implementation of a singly linked list

```
Node a = new Node(5);
Node b = new Node(3);
Node c = new Node(10);
Node d = new Node(12);
```


```
a.next = b;
b.next = c;
c.next = d;
```



Displaying a Linked List



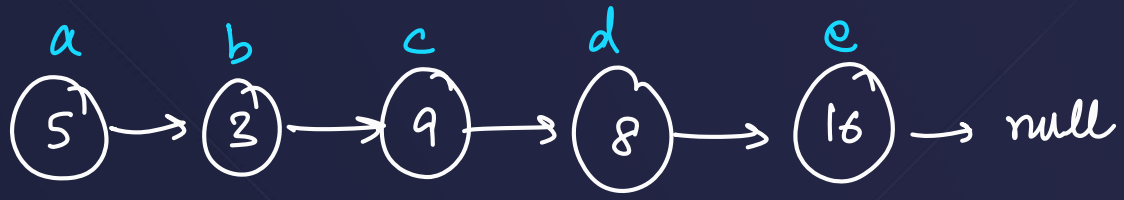

```
Node temp = a;
for(int i=1;i<=5;i++){
    System.out.print(temp.data+" ");
    temp = temp.next;
}
```

temp

Output

5 3 9 8 16

Displaying a Linked List

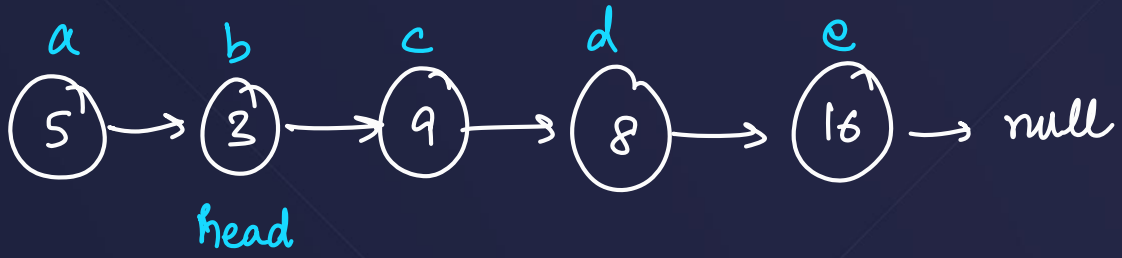



```
while (a != null) {
    Sout (a.data + " ");
    a = a.next;
}
```

Output

5 3 9 8 16

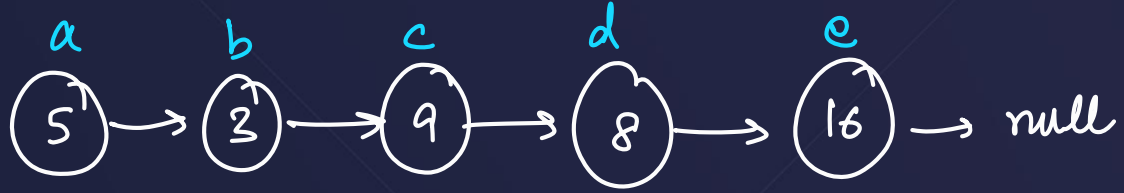
Displaying a Linked List




```
public static void display(Node head){
    while(head!=null){
        System.out.print(head.data+" ");
        head = head.next;
    }
}
```

Displaying a Linked List

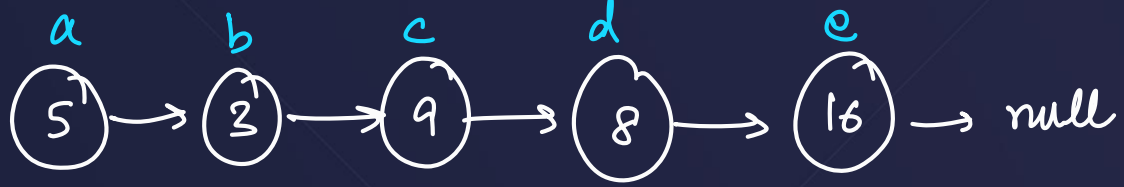
Can we do it recursively?




```
void displayr (Node head){
    if(head == null) return;
    Sout(head.data);
    displayr(head.next);
}
```

Displaying a Linked List

Can we do it recursively?



displayr(a);

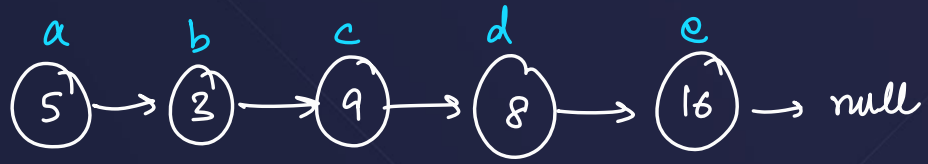
```
public static void displayr(Node head){
    if(head==null) return;
    System.out.print(head.data+" ");
    displayr(head.next);
}
```

Output

5 3 9 8 16

Displaying a Linked List (Reverse)

Can we do it recursively?



display reverse (a) ;

```
public static void displayreverse(Node head){
    if(head==null) return;
    displayreverse(head.next);
    System.out.print(head.data+" ");
}
```

Output

16 8 9 3 5

What will this function do?

```
void func(Node head) {
    if(head == null) return;
    func(head.next);
    System.out.print(head.val + " ");
}
```

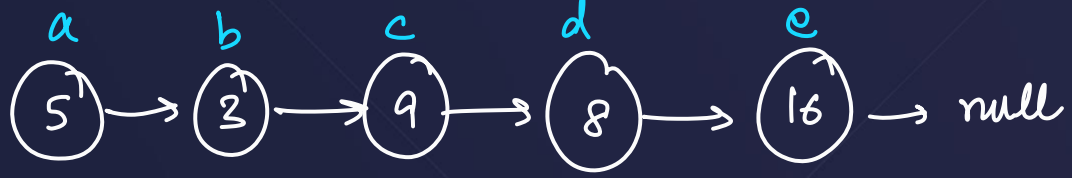
data

```
public static void displayr(Node head){
    if(head==null) return;
    System.out.print(head.data+" ");
    displayr(head.next);
}
```

(a) Print all the elements of the linked list.
(b) Print all the elements except last one.
(c) Print alternate nodes of linked list
(d) Print all the nodes in reverse order

Length method

Implement a method to find out the length of a Linked List (Iterative and Recursive)

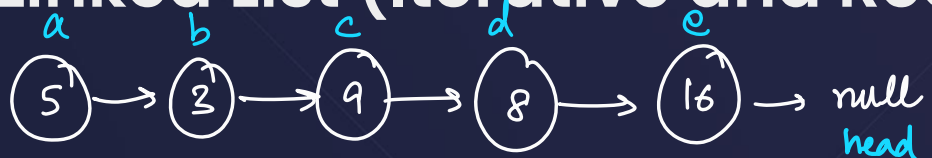



```
int len = length(Node a);

int length(Node a){
    int count = 0;
    while (a != null){
        count ++;
        a = a.next;
    }
    return count;
}
```

Length method

→ if only head is given → O(n)
→ O(1) if ll class is there

Implement a method to find out the length of a Linked List (Iterative and Recursive)



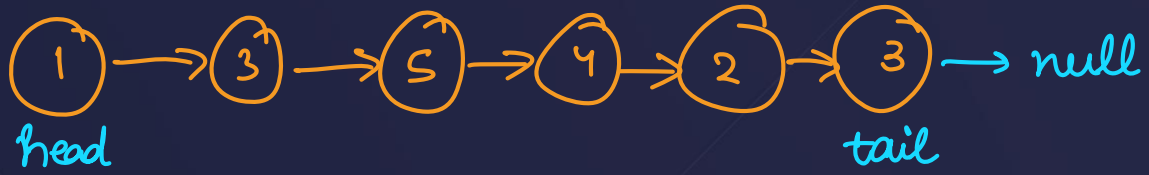
count = 0 1 2 3 4 5

```java
public static int length(Node head){
    int count = 0;
    while(head!=null){
        count++;
        head = head.next;
    }
    return count;
}
```

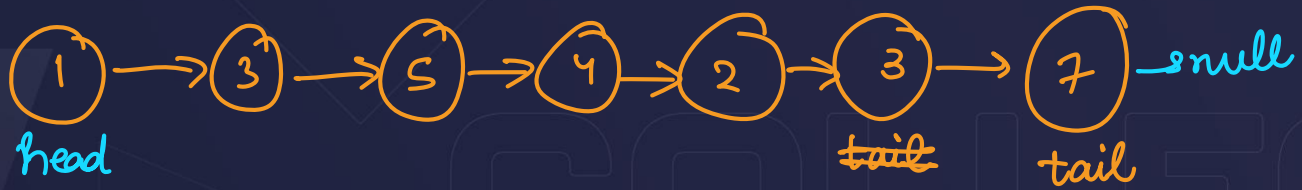
Output

5

Implementation of a singly linked list



ll . insert At End (7) ;



Implementation of a singly linked list

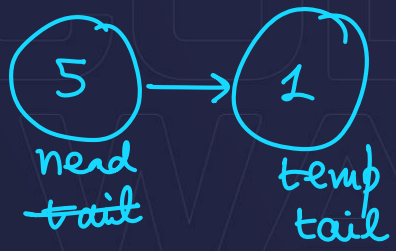
```
ll.insertAtEnd(1);
Node temp = new Node(1);
```

For Empty list

1) if (head == null)

```
head = temp;
tail = temp;
```

For Non-Empty



InsertAtEnd method

O(1) → if tail is given

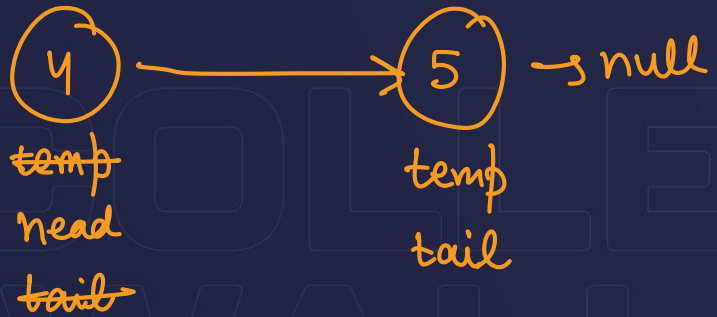
O(n) → if head is given

Implement a method to insert a node at the end of a linked list.

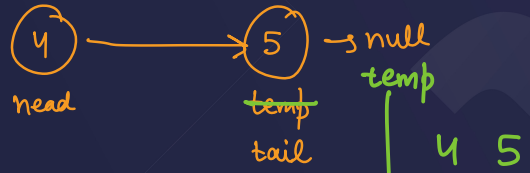
```
void insertAtEnd(int val){
    Node temp = new Node(val);
    if(head==null){
        head = temp;
    }
    else{
        tail.next = temp;
    }
    tail = temp;
}
```

ll.insertAtEnd(4);

ll.insertAtEnd(5);



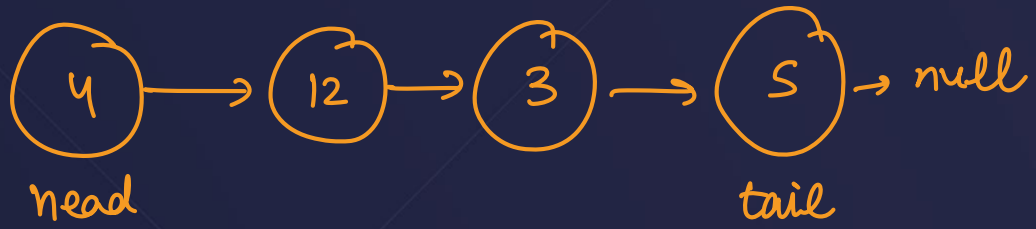
Display in ll class




```
void display(){
    Node temp = head;
    while(temp!=null){
        System.out.print(temp.data+" ");
        temp = temp.next;
    }
}
```

InsertAtBeginning method

Implement a method to insert a node at the start of a linked list.



ll. insertAtHead(5);

InsertAtBeginning method O(1)

Implement a method to insert a node at the start of a linked list.

Empty List :

head == null

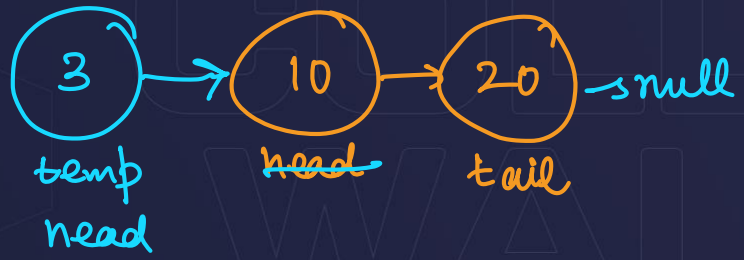
ll.insertAtHead(3);

Non Empty List

ll.insertAtHead(3);

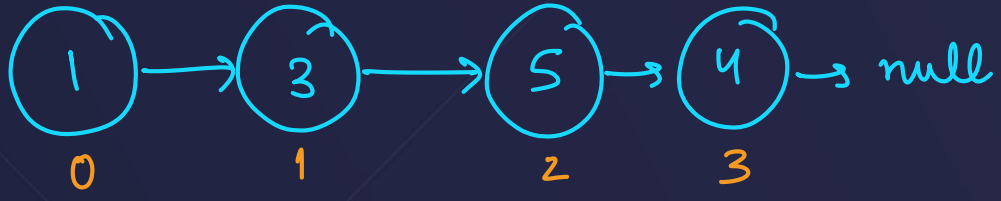
temp.next = head

head = temp;

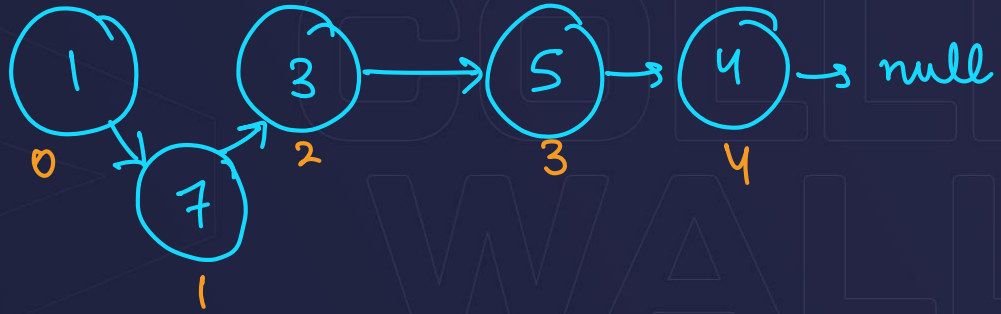


Insert method

Implement a method to insert a node at any given index.



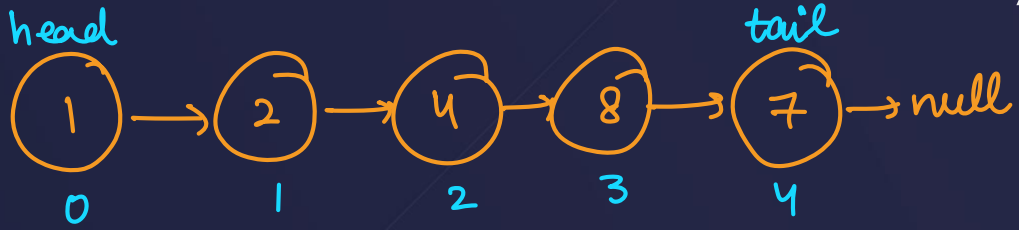
ll.insertAt(1,7);



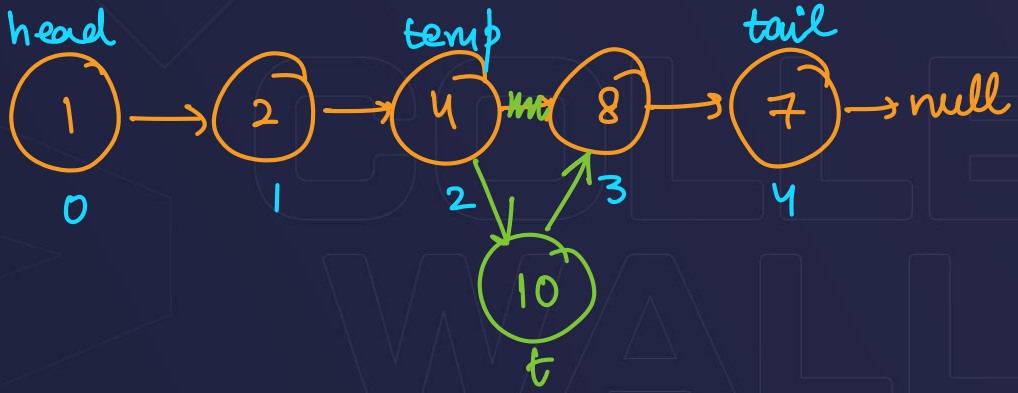
Insert method

T.C. → O(n) S.C. → O(1)

Implement a method to insert a node at any given index.



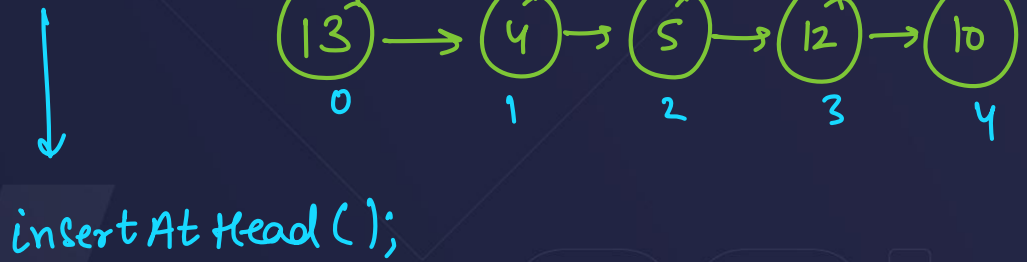
ll.insertAt(3,10);



Insert method

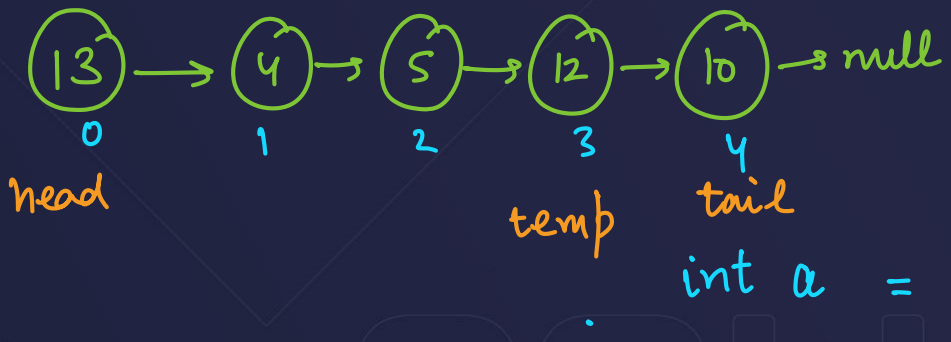
Implement a method to insert a node at any given index.

Case : idx == 0



getElement method → O(n)

Implement a method to return the element at any given index of the linked list.



Evident limitations of Linked List

```
InsertAtEnd(Node head, int val ){
```

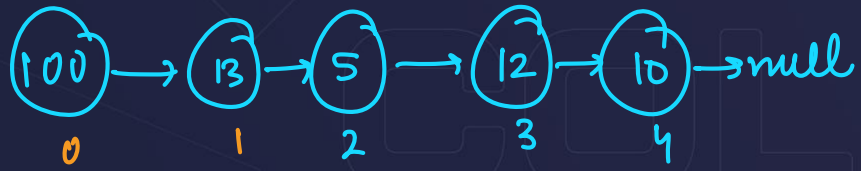
```
insertAtEnd(a, 87);
```

```
}
```

deleteAtIndex method

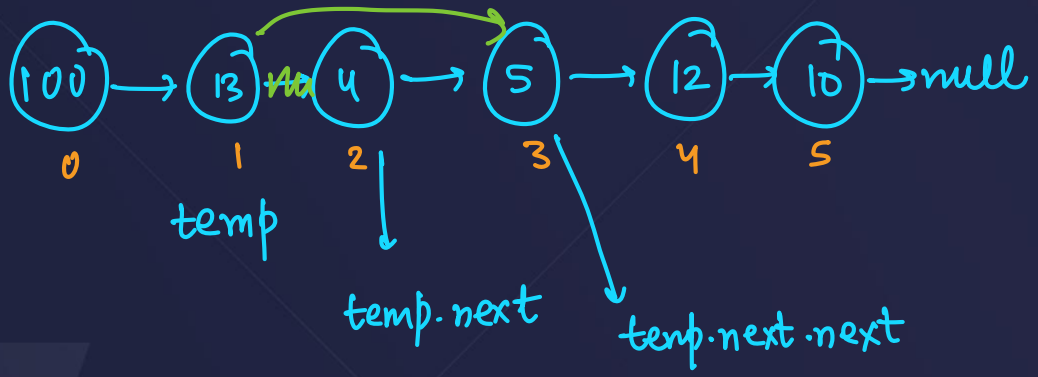
Implement a function to delete a node at a given index

ll.deleteAt(2);



deleteAtIndex method

Implement a function to delete a node at a given index

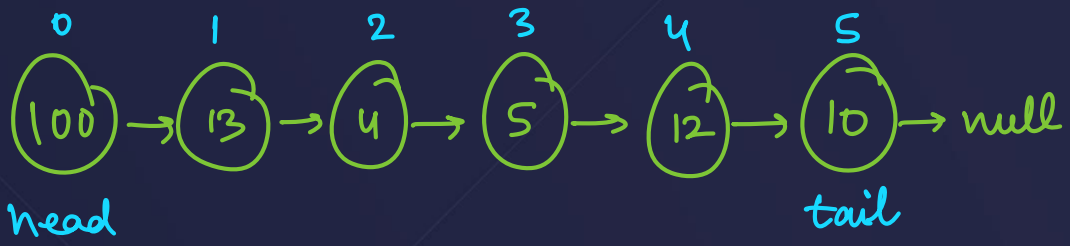


ll.deleteAt(2);

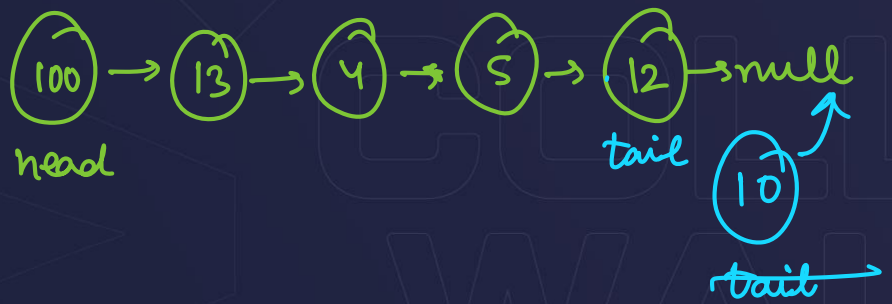
temp.next = temp.next.next;

deleteAtIndex method

Implement a function to delete a node at a given index



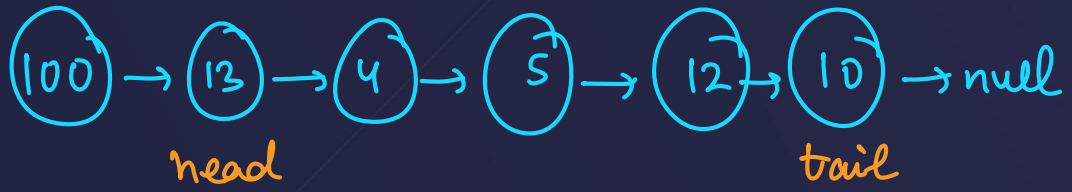
ll.deleteAt(5);



deleteAtIndex method O(n)

Implement a function to delete a node at a given index

if (idx == 0)



ll.deleteAt(0);

```
if (idx == 0)
    head = head.next;
```