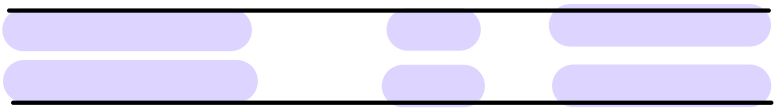
LinkedList 1

Good Evening

* Agenda for Today

01. Basics of Linkedlist
02. Accessing of element
03. Searching for a value
04. Insertion in LL
05. Deletion in LL
06. Reverse a Linkedlist
07. Check if palindrome

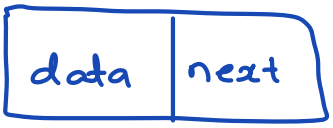
* Arrays Vs Linkedlist



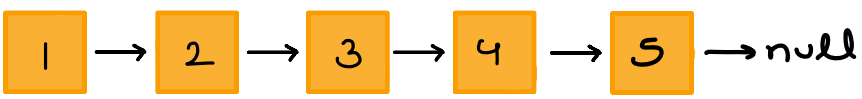
Array → contigous memory allocations

Linkedlist → It can help in utilising all the free memory available

↳ Insertion / Deletion in LL is better as compared to Arrays

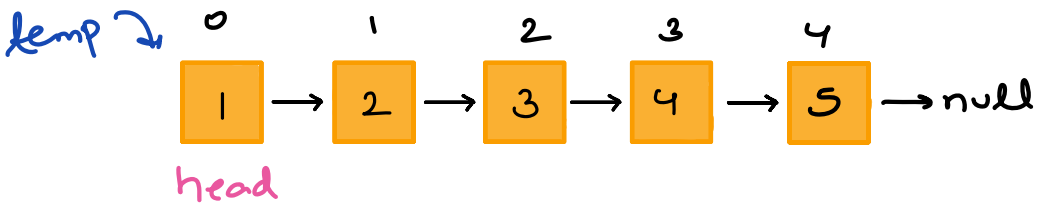



```
class Node {
    int data;
    Node next;

    Node (int d) {
        data = d;
        next = null;
    }
}
```



arr = { 1 2 3 4 5 }

0 1 2 3 4



01. Access a particular index

* Access K^{th} idx element in array = Arr[K] TC:O(1)

```
Node temp = head;
for (i = 1 to K)
|    temp = temp.next;
3
return temp.data;
```

TC: O(K)

01. Never move the head
02. Forward = temp = temp.next;
03. Access data = temp.data;

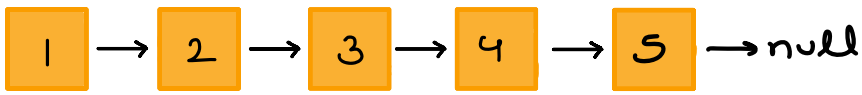
* Search for an ele

Array → linear search → O(n)

↳ Binary Search (only for sorted) → O(log n)

Linkedlist

Search for k=5



head

→ if (head==null) return false; ← 01. Empty LL

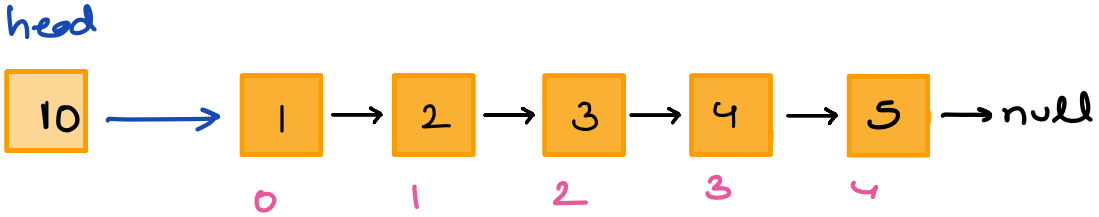
```
Node temp = head;
while (temp!=null){
    if (temp.data == k)
        return true;
    temp = temp.next;
}
return false;
```

TC: O(n)

```
if (head == null) return false:
Node temp = head;
while (temp.next != null)
    if (temp.data == k)
        return true;
    temp = temp.next;
if (temp.data == k) return T
return F
```

Q Insert a value x at k^{th} position (0-based) in the given LinkedList $(0 \le k \le N)$

↳ length of LL



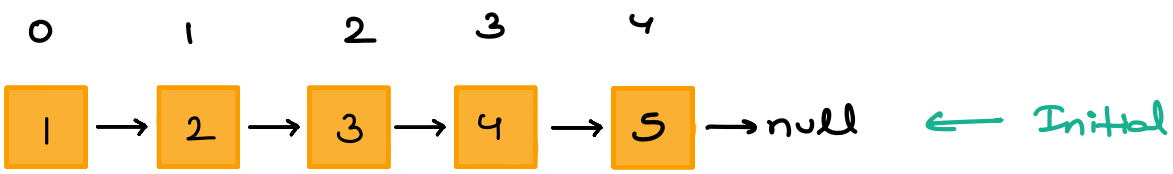
$x = 10$ at $k = 0$

```
Node nn = new Node(10);
```

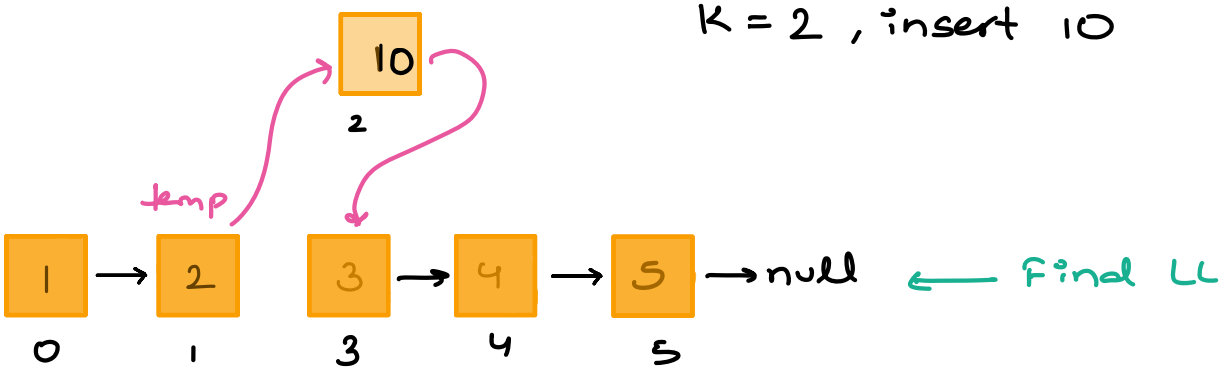
```
if (k==0){
    nn.next = head
    head = nn;
}
else {
    Node temp = head;
    for (i=1; i<=k-1 ; i++){
        temp= temp.next
    }
    nn.next = temp.next
    temp.next = nn;
}
```

TC: O(N)

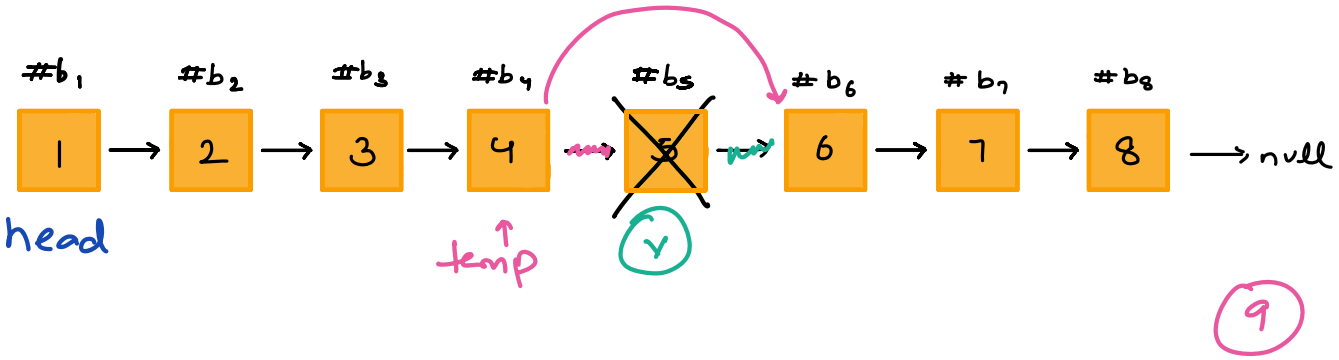
SC: O(1)



K = 2 , insert 10



* Delete the first occurrence of value x in the given LL.



9

```
if (head == null) return head;
Node temp = head;
if (temp.data == x){
    head = head.next;
    return head;
}
while (temp != null){
    if (temp.next != null && temp.next.data == x)
    {
        temp.next = temp.next.next;
        return head;
    }
    temp = temp.next;
}
return head;
```

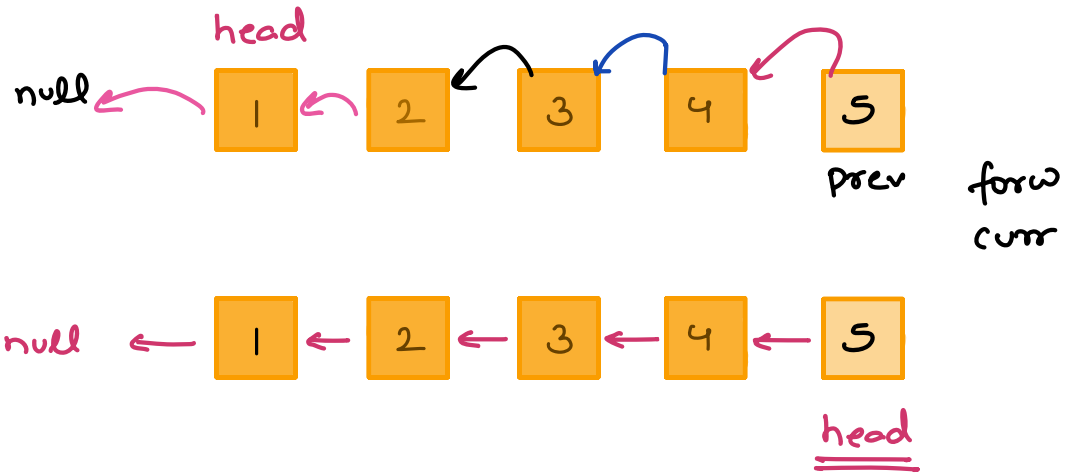
Node v = temp.next

v.next = null

10:14 pm → 10:24 pm

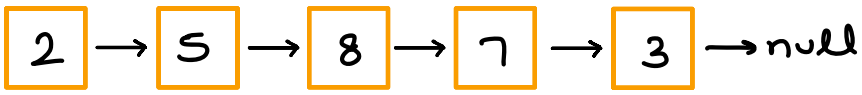
+1

Q Reverse the given linkedlist by updating pointers

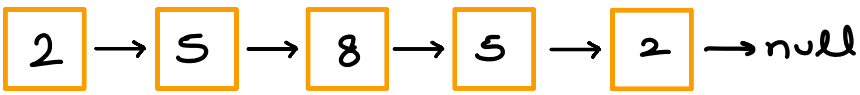



```
Node prev = null

Node curr = head

while ( curr != null ){

    Node forw = curr.next

      curr.next = prev

    prev = curr

    curr = forw

}

head = prev;

return head;
```

Q Check if the given LL is palindrome or not



Ans = false



Ans = true

* BF approach → Create a copy of this LL & reverse it compare if equal or not

TC : O(n)

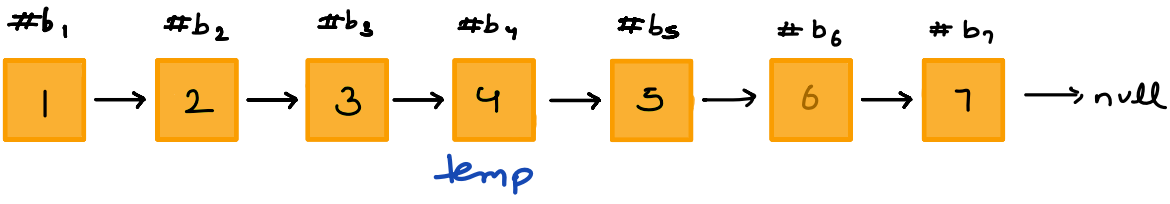
SC : O(n)

Optimise SC : O(1)

01. Middle of LL ⟶ length of LL / 2

02. Reverse second half of LL

03. Compare first half & second half



01. n = 0 , temp = head

```
while (temp != null){
    temp = temp.next;
    n++;
}
```

n = 7 mid = $\frac{7}{2}$ = 3

```
temp = head;
for (i = 1 → mid){
    temp = temp.next;
}
```

i = 1	i ≤ 3
2	2 ≤ 3
3	3 ≤ 3
4	4 ≤ 3 X

* Take temp as head of second LL && reverse it.

* Compare both of them

a b b a → (ab) (ba) compare = 2

[ab c] ba abc ba ✓

ab cba ✓