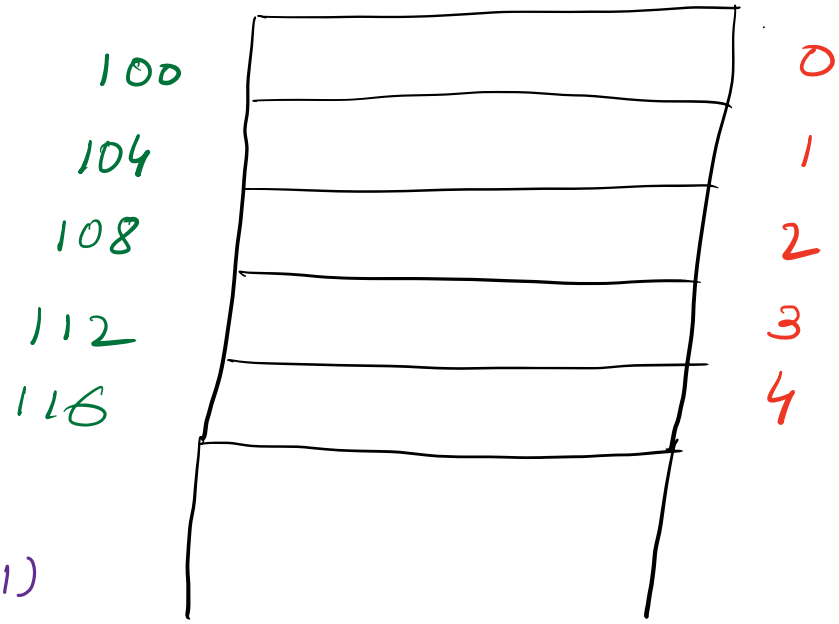
Most commonly used data structure? arrays

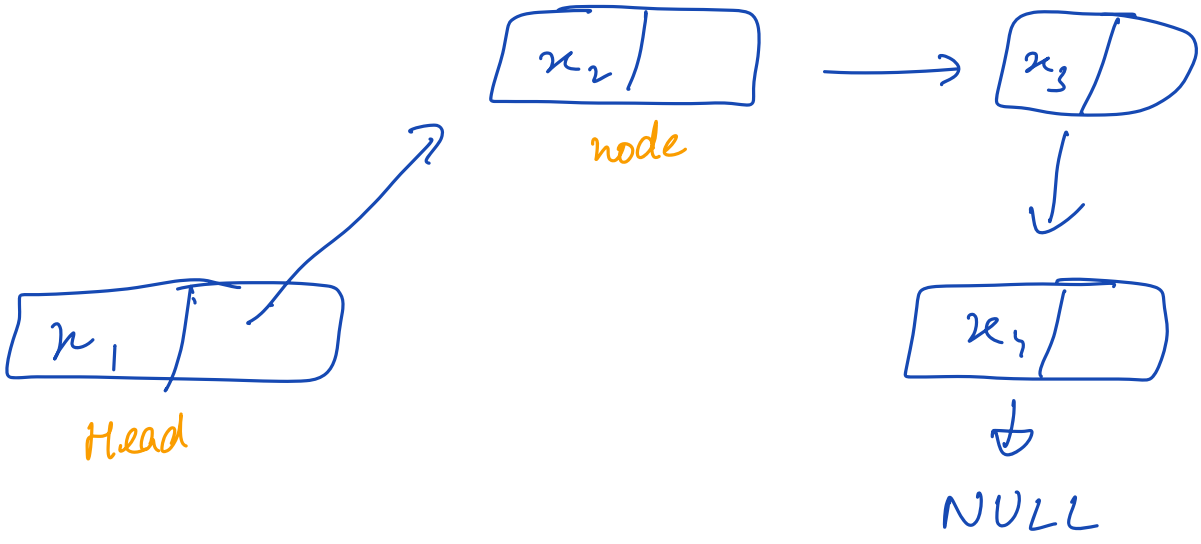
Why? Random elem access is O(1)

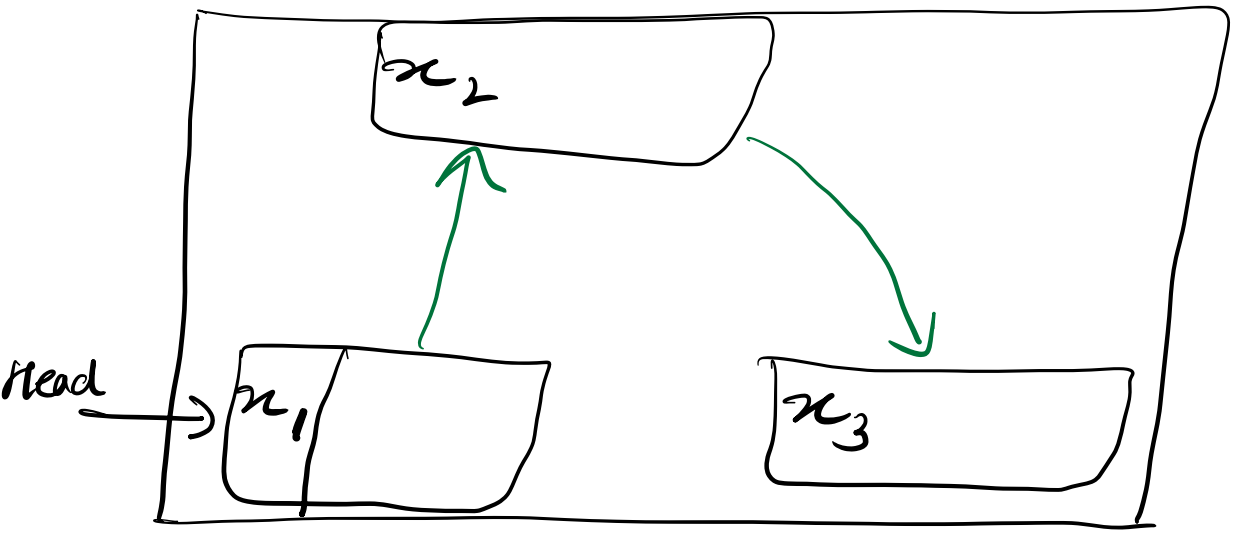


i^{th} index → arr[i]

O(1)

Linked List

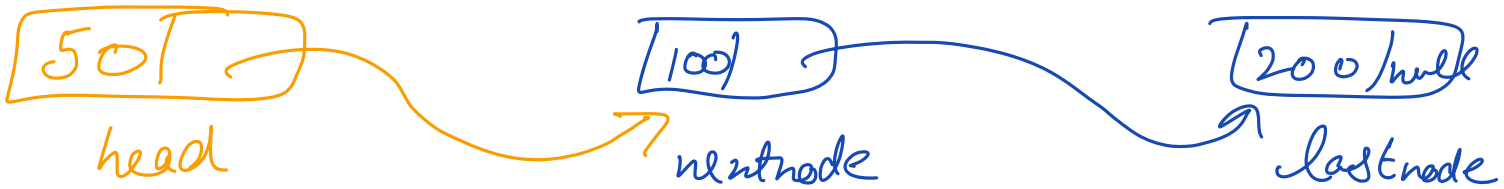




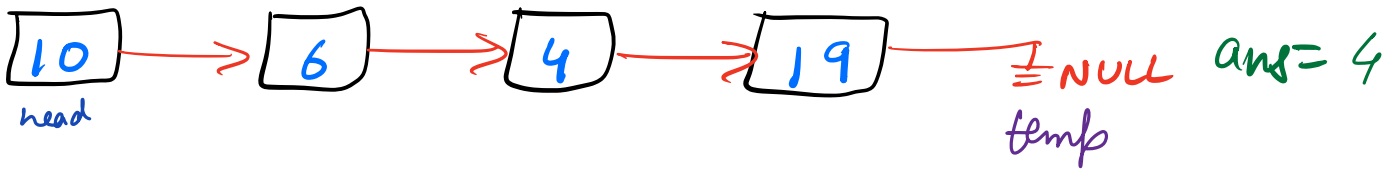
```
class Node {
    int data;
    Node next;

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

```
Node head = new Node(50);
Node nextnode = new Node(100);
head.next = nextnode;
Node lastnode = new Node(200);
nextnode.next = lastnode;
head.next.next = lastnode;
head.next.next.data = 200;
```



Q! Find the size of given Linked List



c = 0 1 2 3 4

```
int size ( Node head ) {
    Node temp = head   // best practice
    int c = 0
    while ( temp != NULL ) {
        c++
        temp = temp.next
    }
    return c
}
```

TC: O(n) SC: O(1)

Deletion from a LL

How to delete highlighted node?

1) If delete node is head

head

head = head.next

return head

2)

goto prev node where

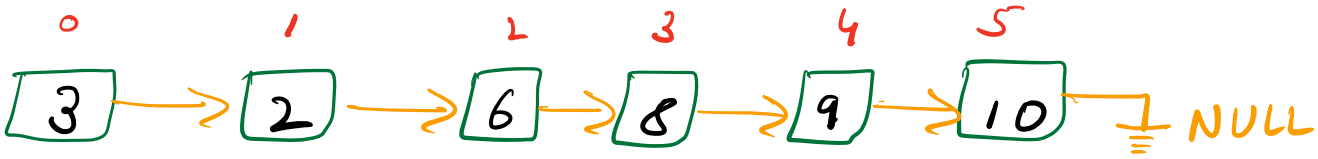
prev.next.value = X

prev.next = prev.next.next

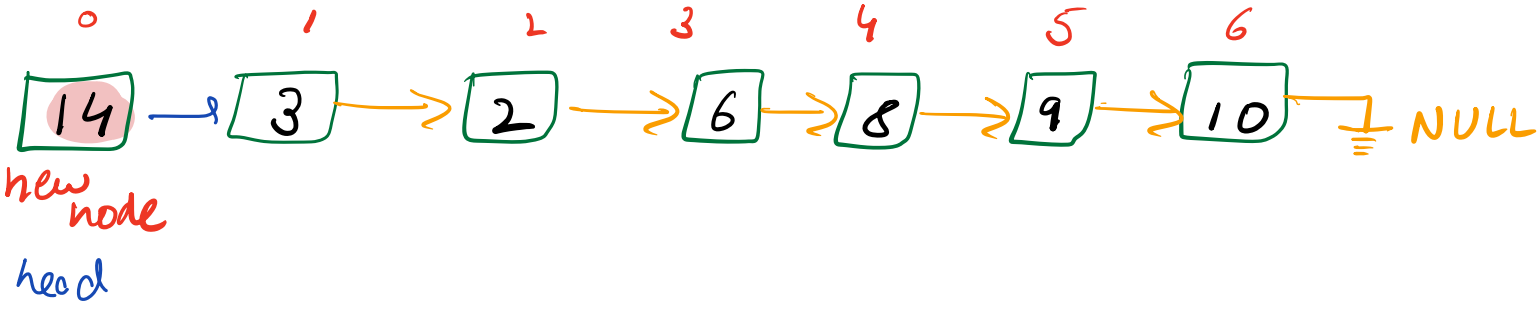
return head

Q3 Insert a node of value x at the K^{th} pos of a Linked List

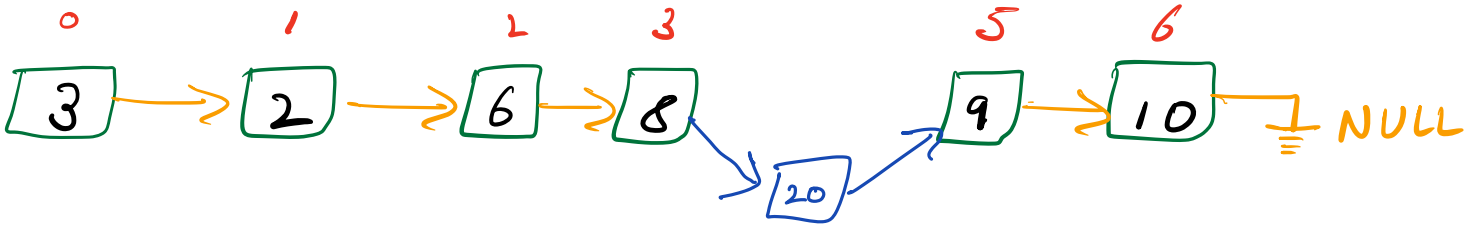
Ex:



Case1 K=0 x = 14

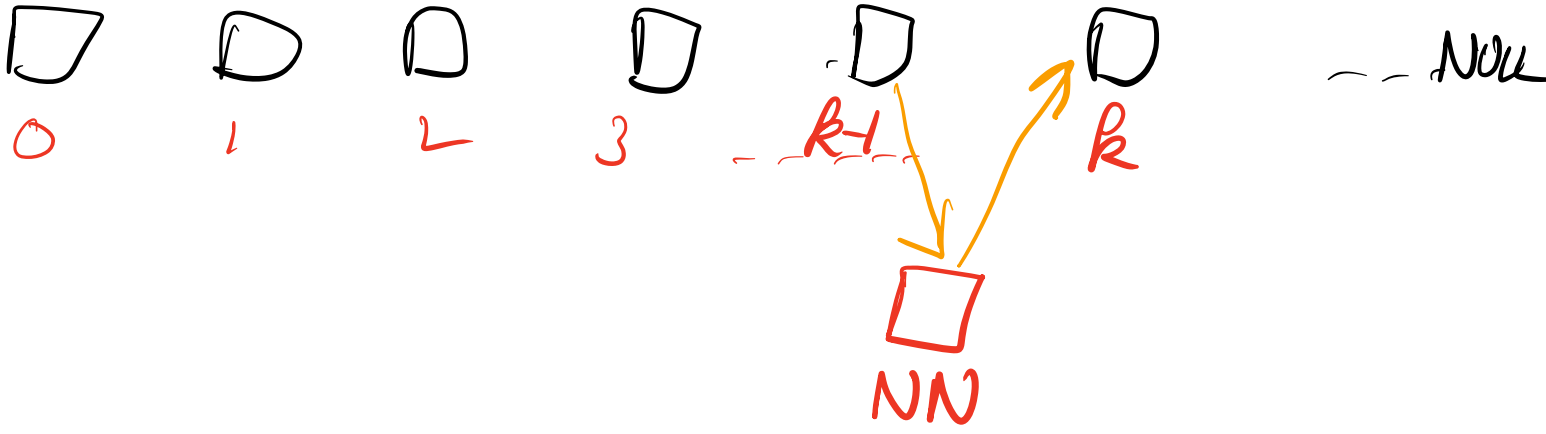


Case 2 K = 4 x = 20



Case 3 K = 7 x = 15

If (K > size of LL) impossible

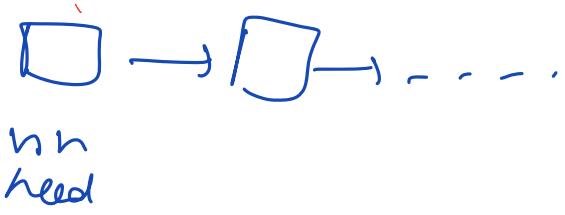


Node insertKpos(int k, int x, Node head){

if(k > size(head))
return head

Node nn = new Node(x)
Node temp = head.

if(k==0){
nn.next = head
head = nn
return head
}



for(i=0; i<k-1; i++){
temp = temp.next
}

temp

// Temp is $k-1^{th}$ node

Node k^{th} node = temp.next

temp.next = nn

nn.next = k^{th} node

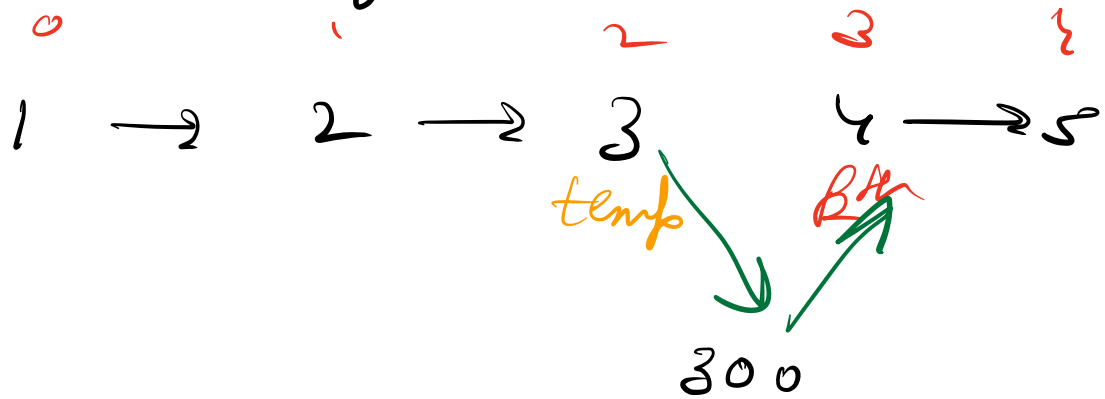
return head

}

TC: O(N)

SC: O(1)

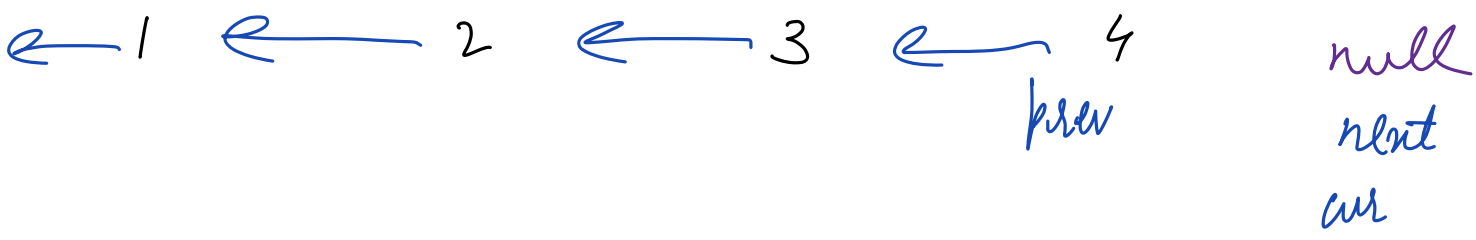
Dry run



Q1 Reverse the L.L

1 → 2 → 3 → 4

4 → 3 → 2 → 1



Code

```
if (head == null || head.next == null)
        return head
cur = head          prev = null
while (cur != null) {
    nextnode = cur.next
    cur.next = prev
    prev = cur
    cur = nextnode
}
head = prev
return head
```

Q Check if LL is palindrome

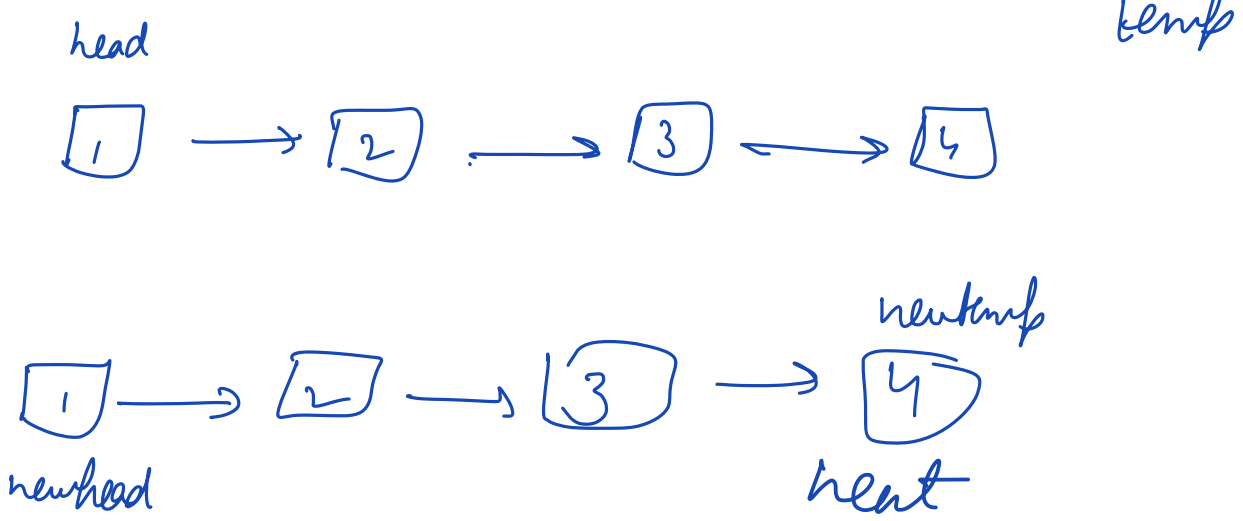
$2 \rightarrow 5 \rightarrow 8 \rightarrow 5 \rightarrow 2$ ans = true

Idea 1 Create a copy of LL, reverse it & compare

TC: O(n) SC: O(n)

1) Create copy

```
Node copy ( Node head ){
    Node temp = head
    Node newhead = new Node ( temp.data)
    newtemp = newhead
    temp = temp.next
    while ( temp != null ){
        Node nextnode = new Node ( temp.data)
        newtemp.next = next node
        newtemp = newtemp.next
        temp = temp.next
    }
    return newhead
}
```



head$_1$ head$_2$

2) Reverse the copy

reverse (head$_2$)

3) See if both LL are same

```
bool isSame ( Node head1 , Node head2 ) {
    temp1 = head1          temp2 = head2
    while ( temp1 != null ) {
        if ( temp1.data != temp2.data )
            return false
        temp1 = temp1.next
        temp2 = temp2.next
    }
    return true
}
```

{done}