Agenda

25th Dec, 1st Jan

1) Understanding Linked List
2) LL printing and size
3) Insert in LL
4) Delete from LL
5) Reverse a LL
6) Check LL is palindromic or not (only steps)

27th Dec, 29th Dec, 3rd Jan → LL

5th Jan onwards → merged classes (MWF)

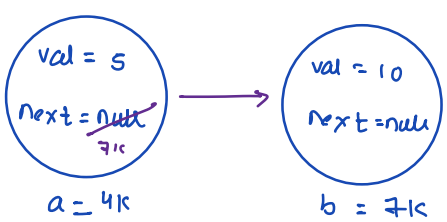
dependency → basics of classes & objects.

What is a LinkedList

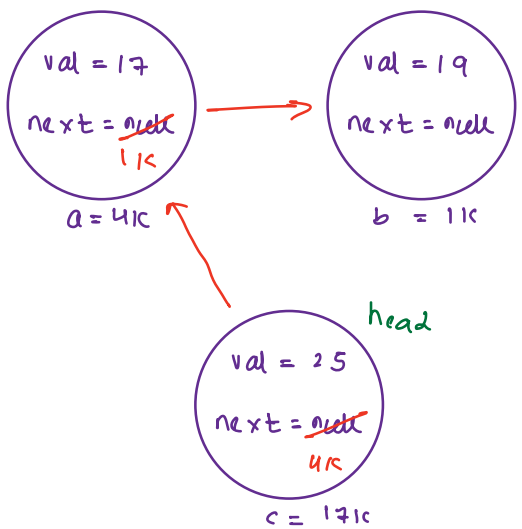
↳ Collection of Nodes

```
class Node {
    int val;
    Node next;

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

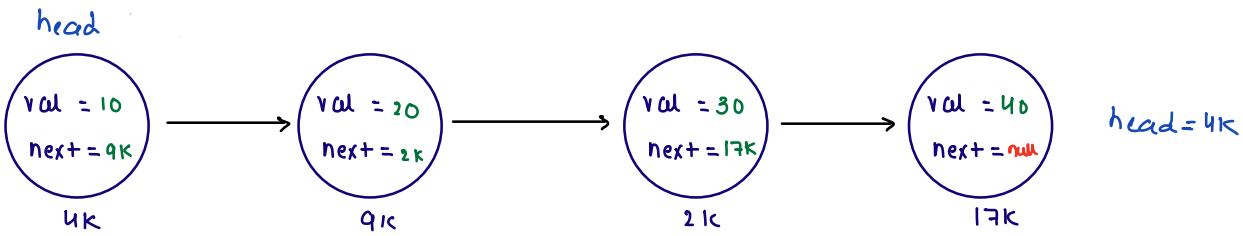
```
Node a = new Node (5);
Node b = new Node (10);

a.next = b;
```



```
class Node {
    int val;
    Node next;

    Node (int val) {
        this.val = val;
    }
}

main ( ) {
    Node a = new Node(17);
    Node b = new Node(19);
    Node c = new Node(25);
    //connections
    c.next = a;
    a.next = b;
}
```



Q.1 Given head of a LL, print the LL.



~~9K 2K 17K~~ null

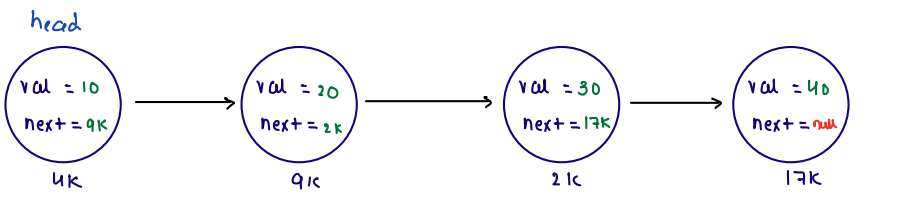
temp = ~~4K~~;

10 20 30 40

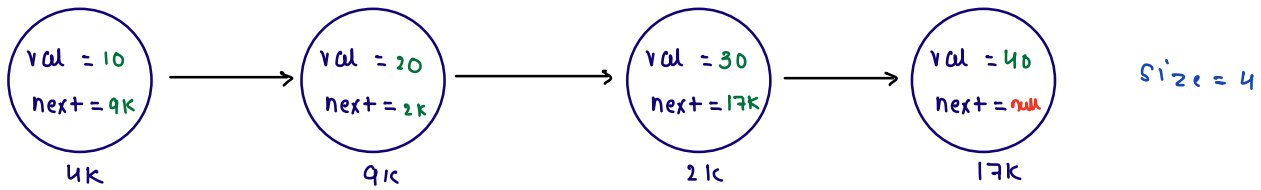
```
Node temp = head;
while (temp != null) {
    SOP (temp.val);
    temp = temp.next;
}
```

```
void printLL (Node head) {
    Node temp = head;
    while (temp != null) {
        SOP (temp.val + " ");
        temp = temp.next;
    }
}
```

10 20 30 40

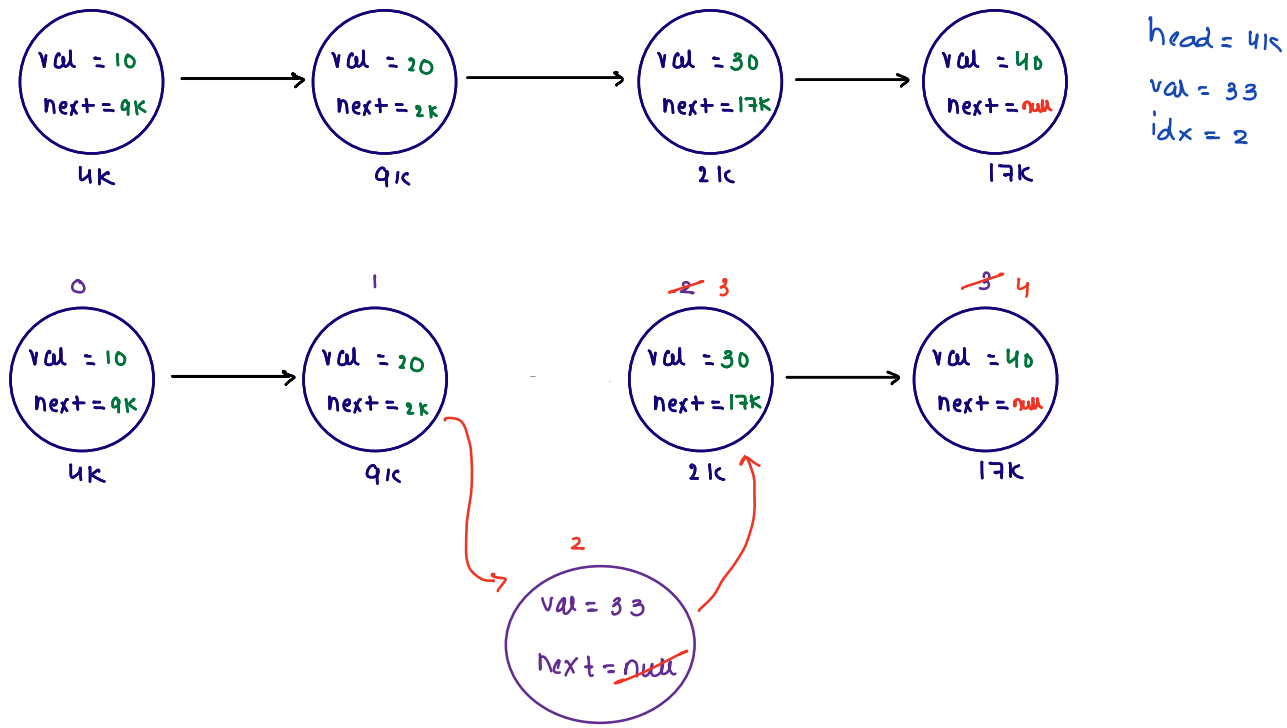


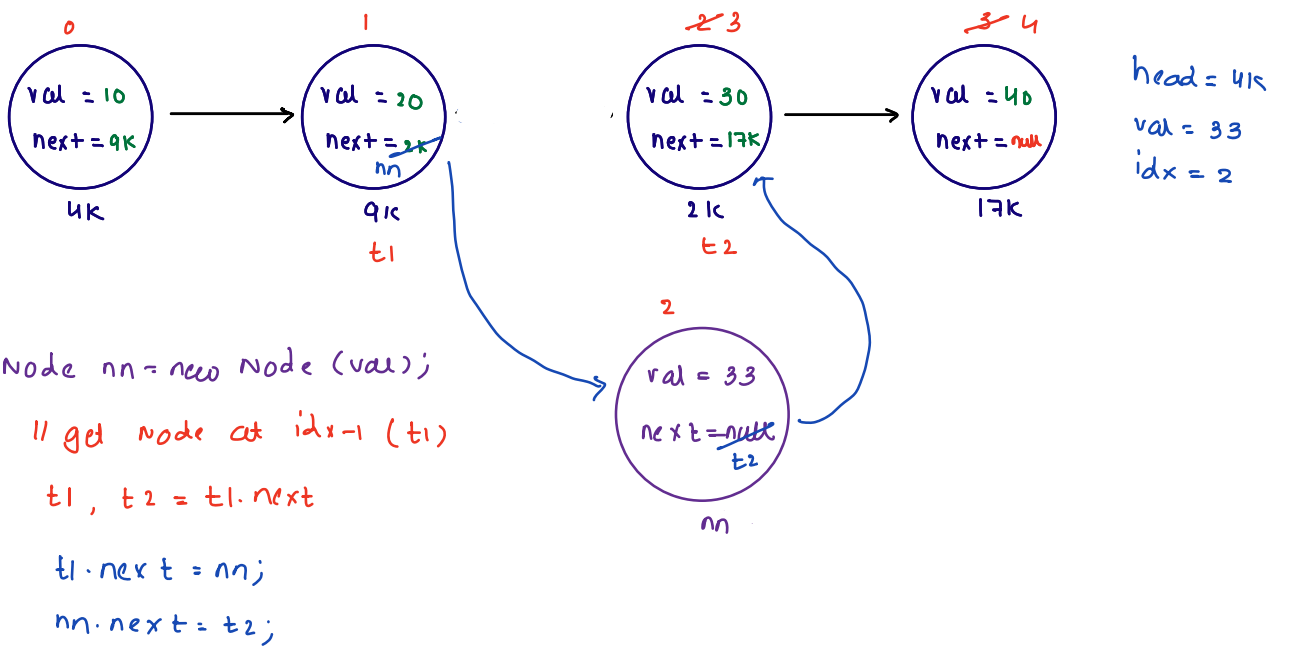
Q.2 Given head of a LL, find its size.



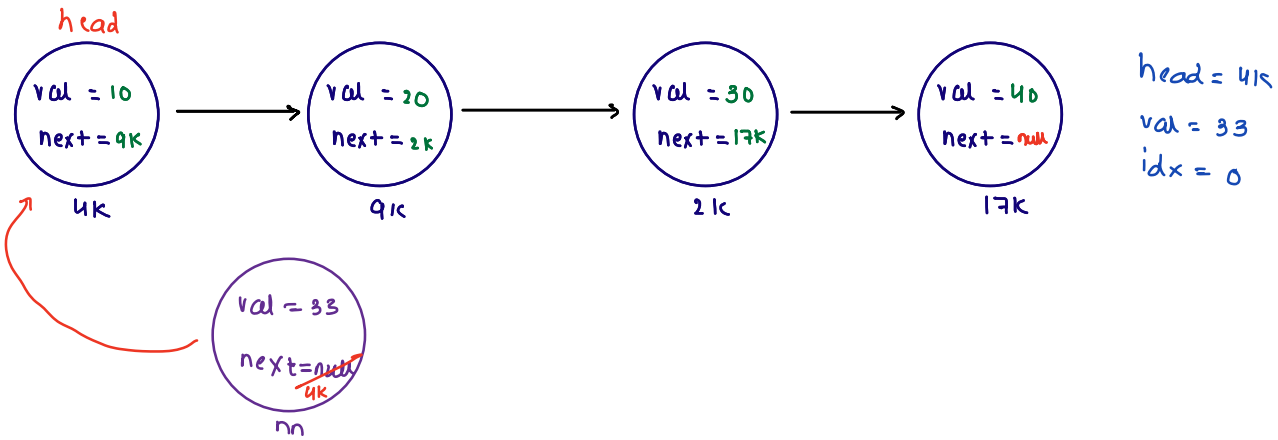

```
int size(Node head) {
    Node temp = head;
    int count = 0;
    while (temp != null) {
        count++;
        temp = temp.next;
    }
    return count;
}
```

Q.3 Given head of a LL, insert node of given value at a particular given index and return head of updated LL. (0 based indexing)






```
Node nn = new Node (val);
// get node at idx-1 (t1)
t1 , t2 = t1.next
t1.next = nn;
nn.next = t2;
```



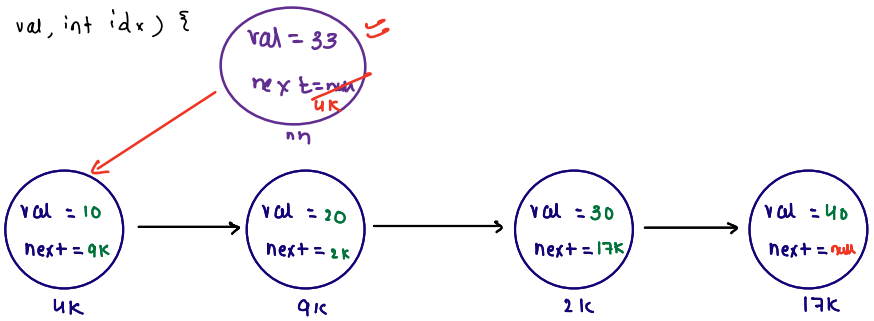
head = 4K
val = 33
idx = 0

```
Node nn = new Node (val);
nn.next = head;
return nn;
```

```
Node insertInLL(Node head, int val, int idx) {
    Node nn = new Node(val);
    if (idx == 0) {
        nn.next = head;
        return nn;
    }
    else {
        //get node at idx-1
        Node t1 = head;
        for (int i=0; i<idx-1; i++) {
            t1 = t1.next;
        }
        Node t2 = t1.next;
        t1.next = nn;
        nn.next = t2;
        return head;
    }
}
```

idx-1 times

```
Node insertInLL ( Node head, int val, int idx ) {

    Node nn = new Node (val);

    if (idx == 0) {
        nn.next = head;
        return nn;
    }
    else {
        //get node at idx-1
        Node t1 = head;
        for (int i=0; i< idx-1; i++) {
            t1 = t1.next;
        }
        Node t2 = t1.next;
        t1.next = nn;
        nn.next = t2;
        return head;
    }
}
```



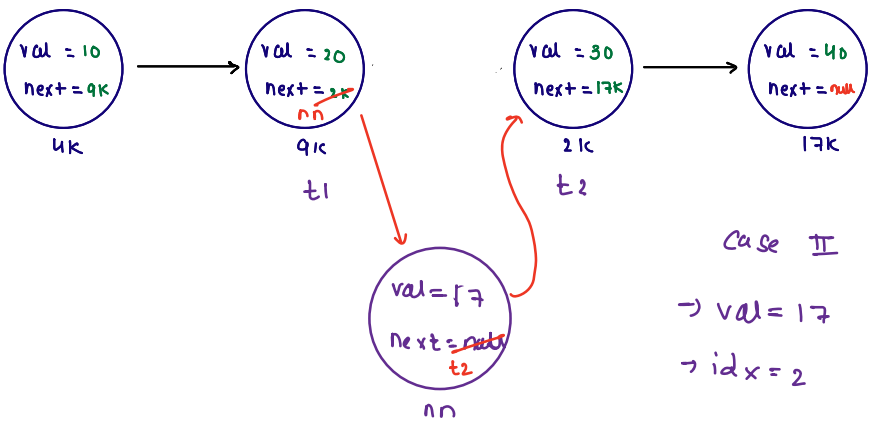
idx-1 times

Case I

→ val = 33

→ idx = 0

return nn

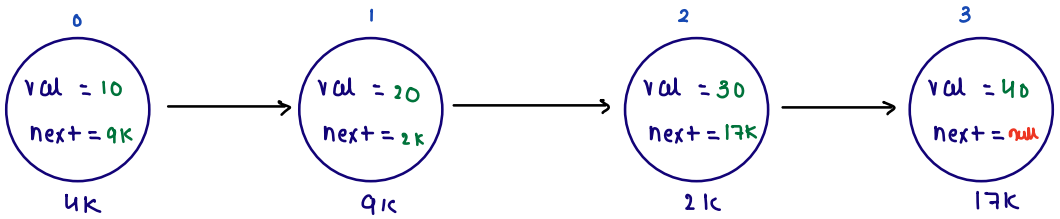


Case II

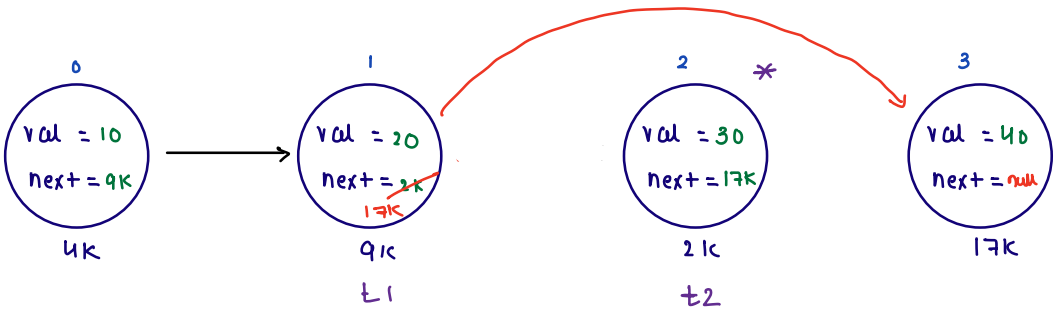
→ val = 17

→ idx = 2

Q.4 Given head of a LL and index, remove node present at this particular index from LinkedList and return head of updated LL.



head = 4K
idx = 2

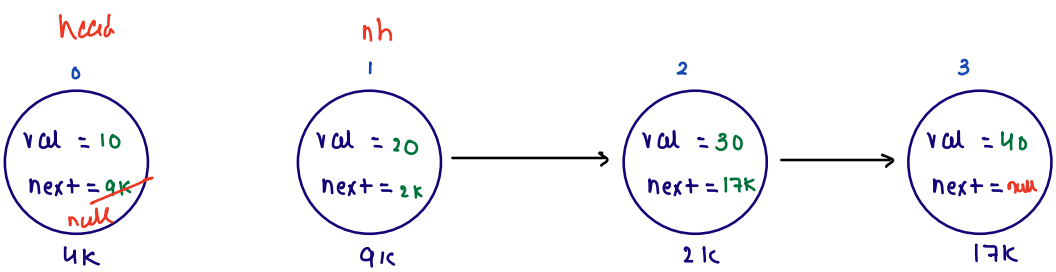


// get node at idx-1 (t1)

t1, t2 = t1.next

→ t1.next = t2.next

→ t2.next = null (optional)



head = 4K
idx = 0

Node nh = head.next;

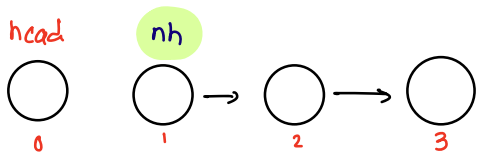
head.next = null;

return nh;

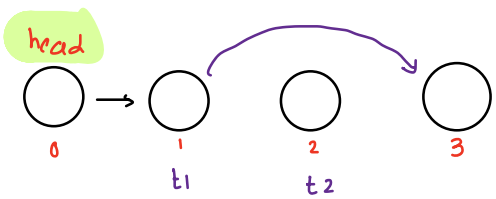
Or

return head.next;

```
Node DeleteFromLL ( Node head, int idx) {

    if ( idx == 0) {
        Node nh = head.next;
        head.next = null;
        return nh;
    }
    else {
        //get node at idx-1
        Node t1 = head;
        for (int i=0; i< idx-1; i++) {
            t1 = t1.next;
        }
        Node t2 = t1.next;
        t1.next = t2.next;
        t2.next = null;
        return head;
    }

}
```



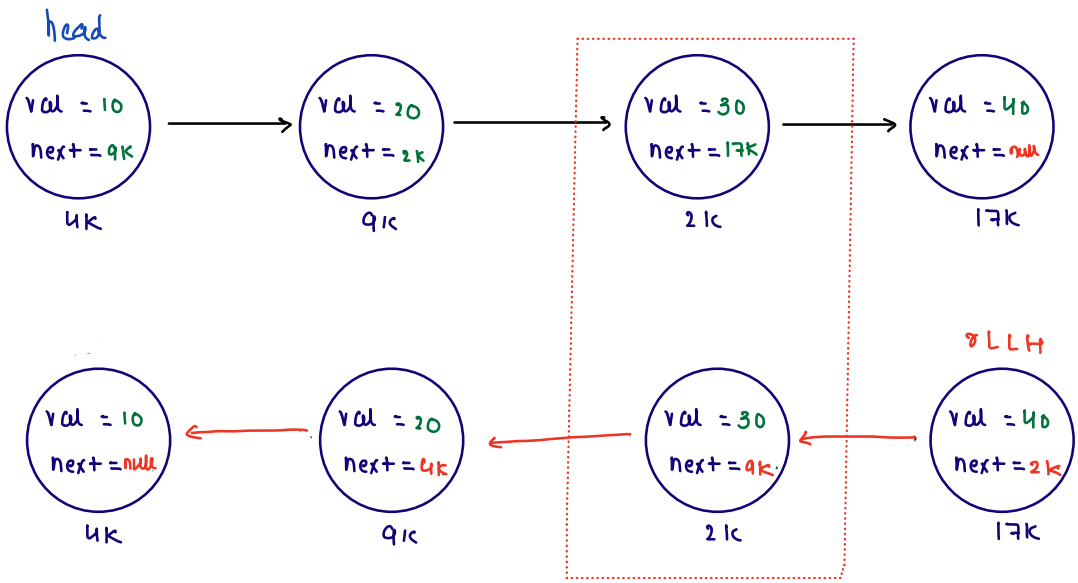
idx=0

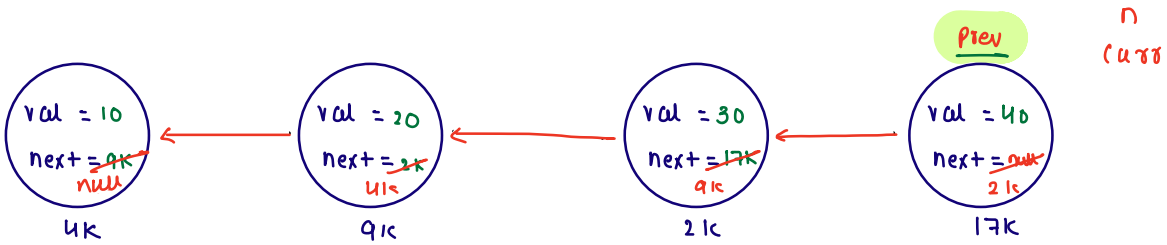


idx=2

Q.5 Given head of LL, reverse it and return head of reversed LL.

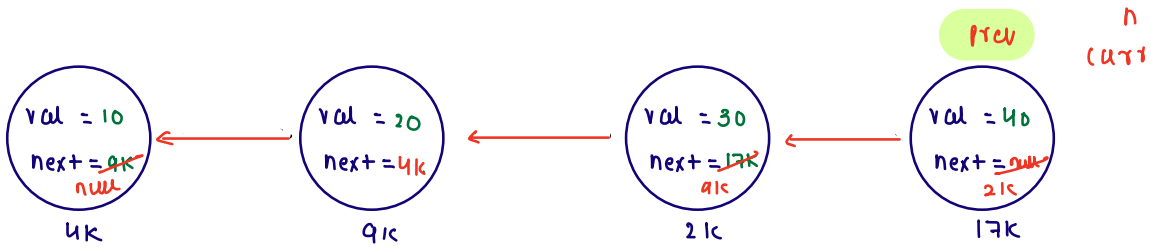
note: Extra space is not allowed.





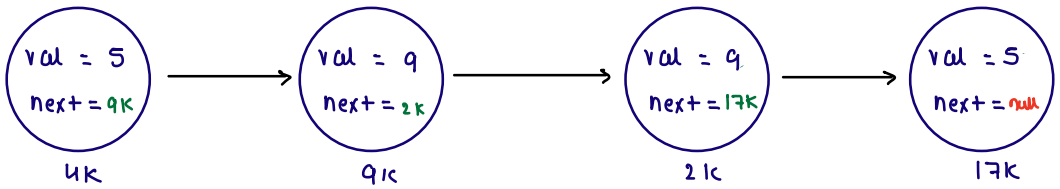

```
while (curr != null)
    Node n = curr.next;
    curr.next = prev;
    prev = curr;
    curr = n;

return prev;
```

```
Node reverseLL ( Node head) {
    Node curr = head, prev = null;
    while ( curr != null) {
        Node n = curr.next;
        curr.next = prev;
        prev = curr;
        curr = n;
    }
    return prev;
}
```

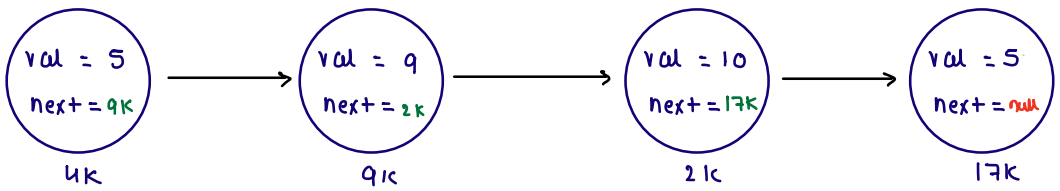


Q.6 Given head of a LL, check if it is palindromic or not.

↳ reverse LL



True

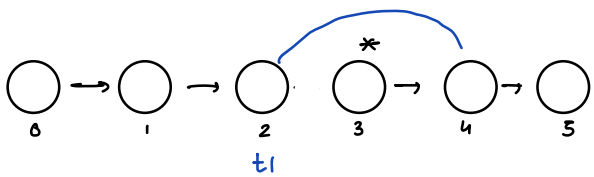


False

Doubts

```
Node DeleteFromLL ( Node head, int idx) {

    if ( idx == 0) {
        Node nh = head.next;
        head.next = null;
        return nh;
    }
    else {
        //get node at idx-1
        Node t1 = head;
        for (int i=0; i< idx-1; i++) {
            t1 = t1.next;
        }
        Node t2 = t1.next;
        t1.next = t2.next;
        t2.next = null;
        return head;

    }
}
```

OR `t1.next = t1.next.next;`



idx = 3