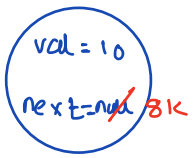
What is Linked list : Collection of Nodes

```
class Node {
    int val;
    Node next;
    Node (int a) {
        val = a;
    }
}
```

```
Node n1 = new Node (10);
Node n2 = new Node (20);
```



4K

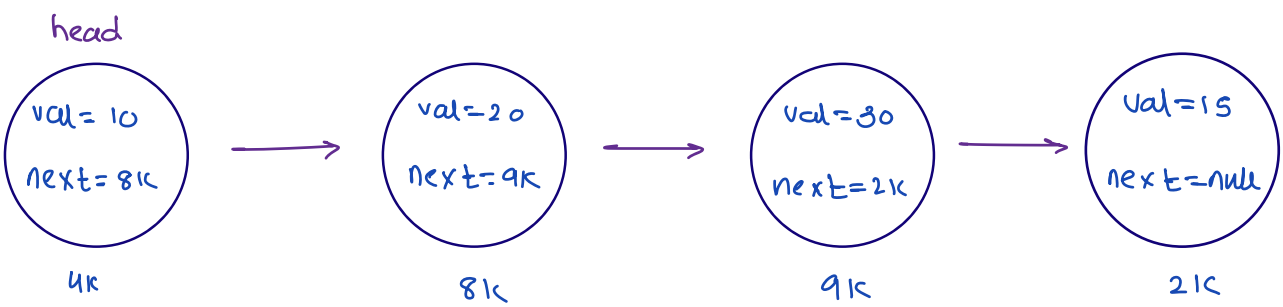
(n1)

8K

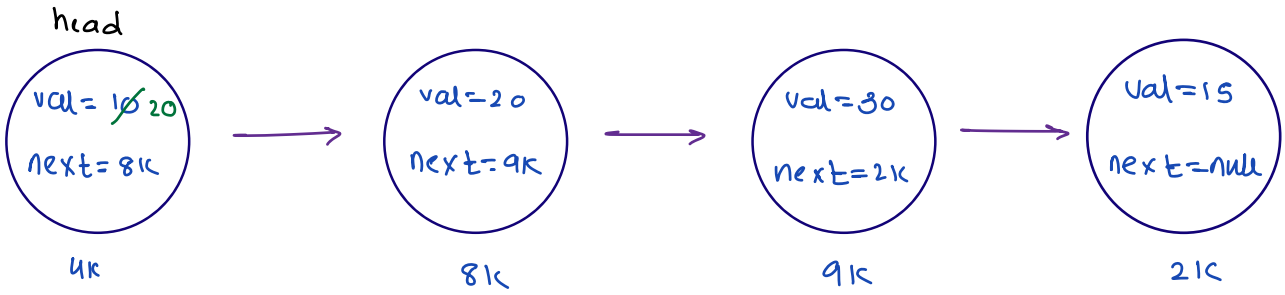
(n2)

n1.next = n2

4K.next 8K



head => 4K



head = 4K temp = 4K

Node temp = head ;

temp . val = temp . next . val ;

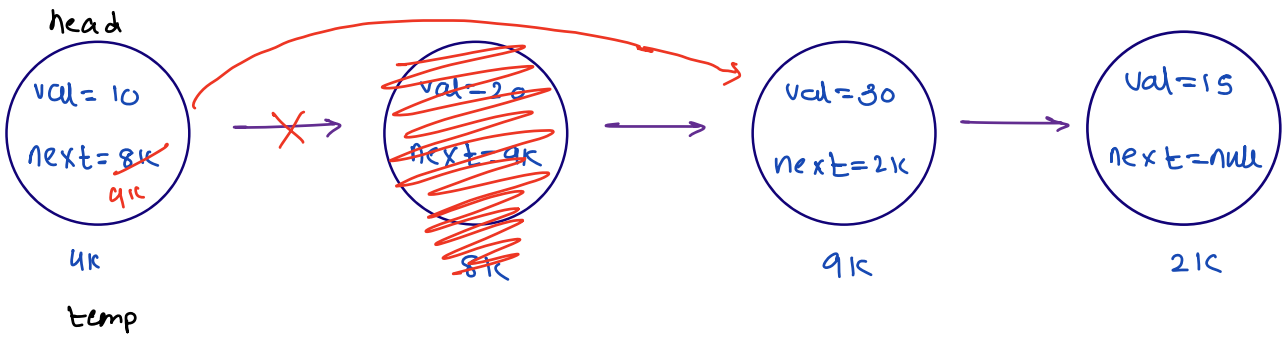
4K . next

8K . val = 20

temp = 4K

temp = temp . next . next ;

temp = 9K ; { just a temporary variable getting changed }



head = 4K

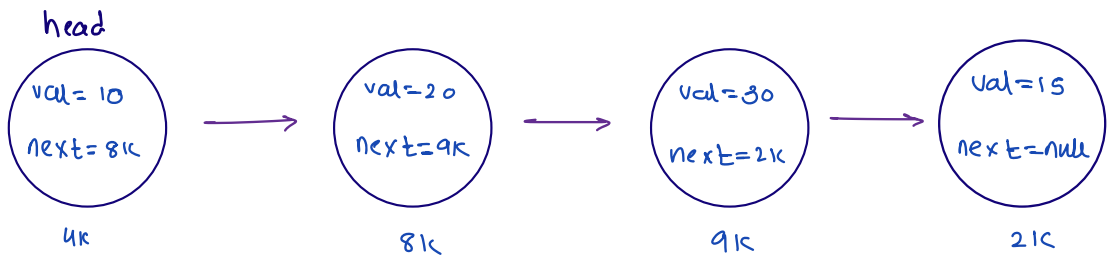
temp = 4K

```
Node temp = head;

temp.next = temp.next.next;
```

4K.next = 9K

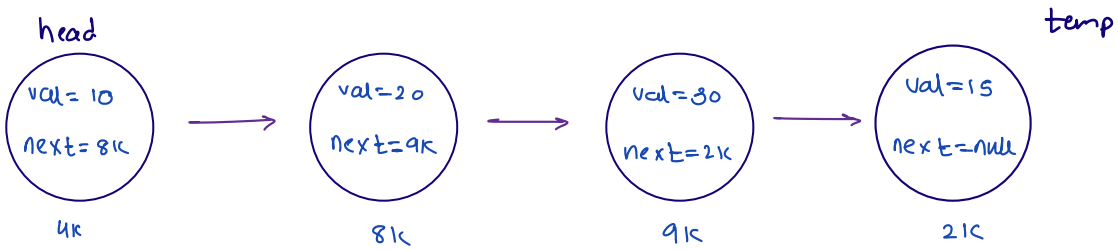
Q.1 Given head of linked list, print the Linked List.



O/P: 10 20 30 15 head = 4K

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

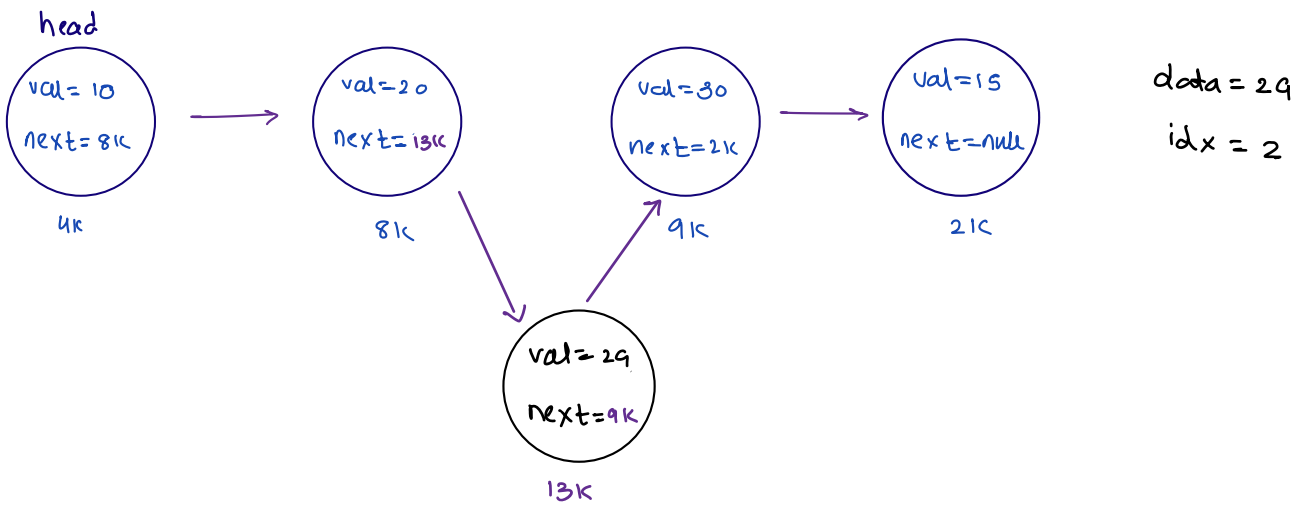
temp	
4K	10
8K	20
9K	30
2K	15
null	



Q-2 Add node at a particular index. {0 based indexing}

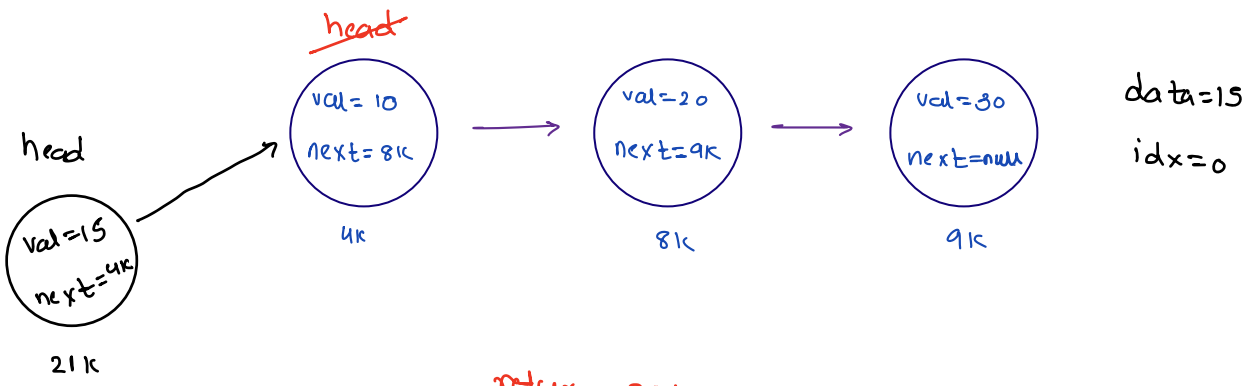
Input: head (head of given LL)

data } add a node with val=data at idx in
idx } given Linked List and return head of linkedlist.

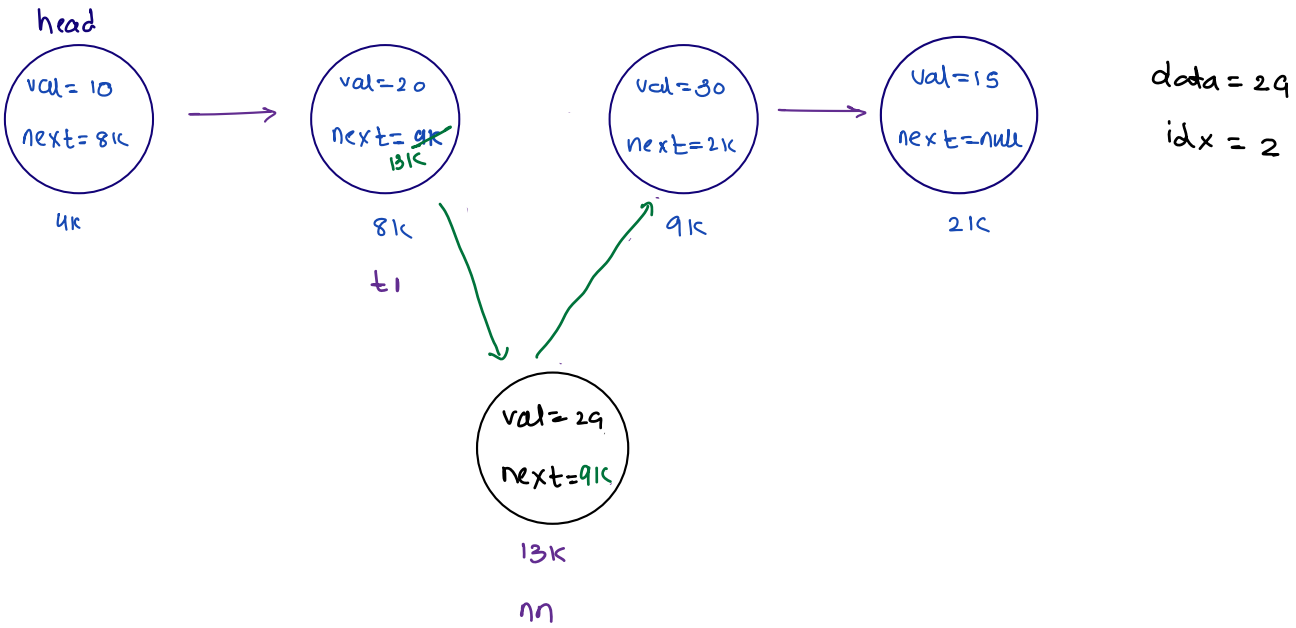


edge case: when idx = 0

{head will change}



return 21K;



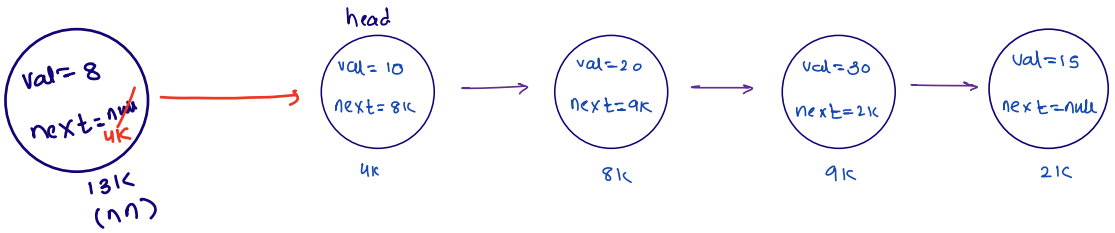

```
Node nn = new Node(data);
//get node present at idx-1
Node t1 ......

nn.next = t1.next;

t1.next = nn;
```

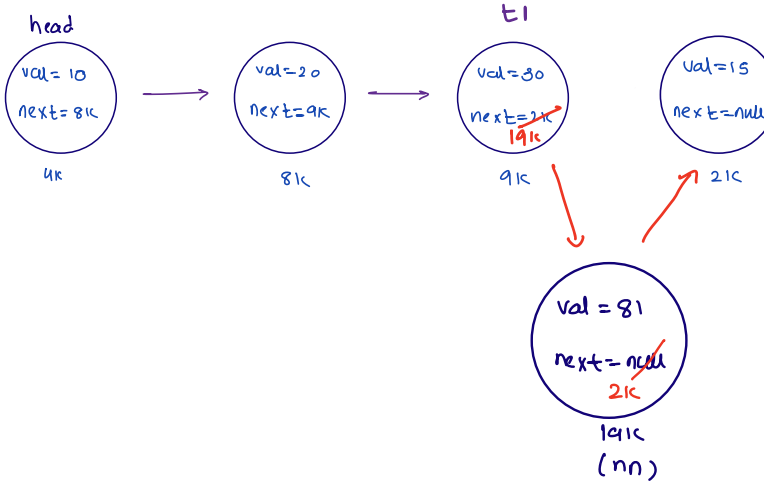
```
Node add (Node head, int data, int idx) {
    Node nn = new Node(data);
    if (idx == 0) {
        nn.next = head;
        return nn;
    }
    else {
        // find node present at idx-1
        Node t1 = head;
        for (int k=1; k<=idx-1; k++) {
            t1 = t1.next;
        }
        nn.next = t1.next;
        t1.next = nn;
        return head;
    }
}
```

Dry run



eg1 { head = 4K, data = 8, idx = 0

```
Node add (Node head, int data, int idx){
    Node nn = new Node(data);
    if (idx == 0) {
        nn.next = head;
        return nn;
    }
    else {
        // find node present at idx-1
        Node t1 = head;
        for (int k=1; k<=idx-1; k++) {
            t1 = t1.next;
        }
        nn.next = t1.next;
        t1.next = nn;
        return head;
    }
}
```

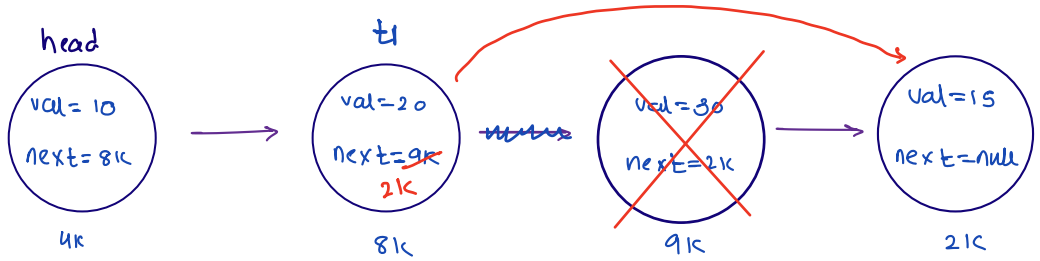


eg2 { head = 4K, data = 81, idx = 3

Q-3 Delete node from a particular idx.

input : head (head of given LL)

idx (remove node from this index and return head of the LL)

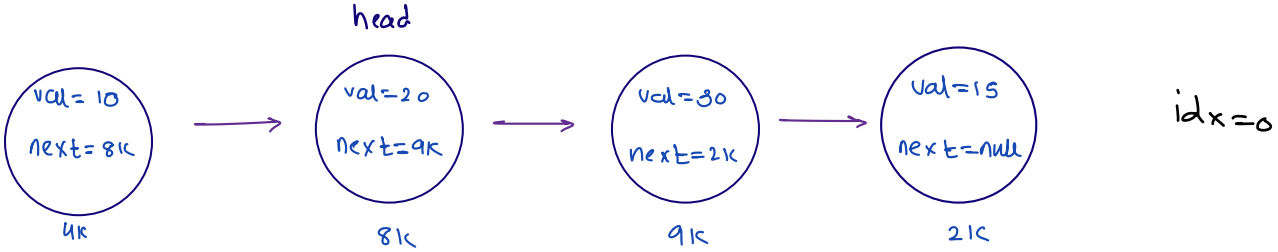


// get node present at idx-1

Node tl ....

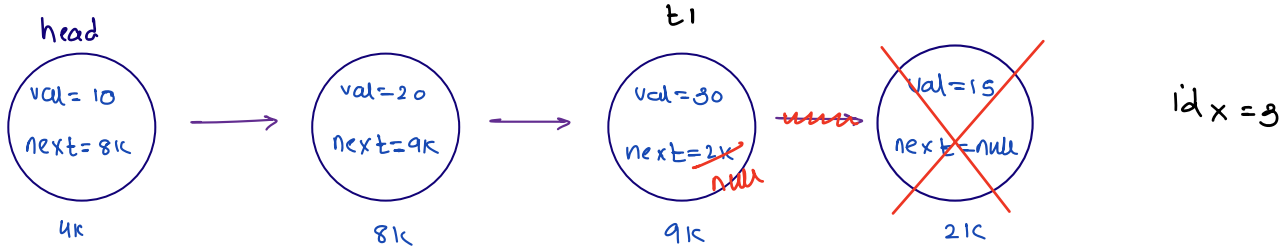
tl.next = tl.next.next

edge case, when idx=0



head = head.next;

return head;

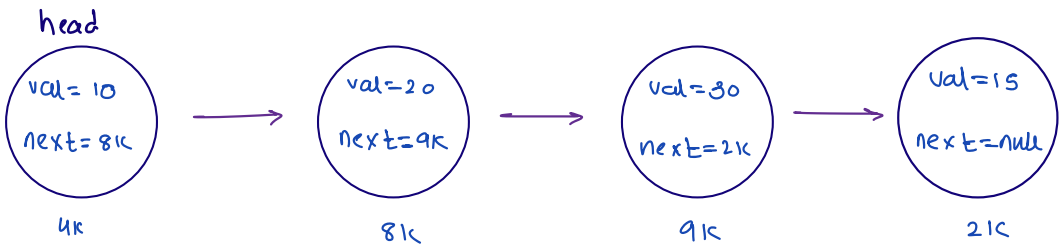


t1.next = t1.next.next

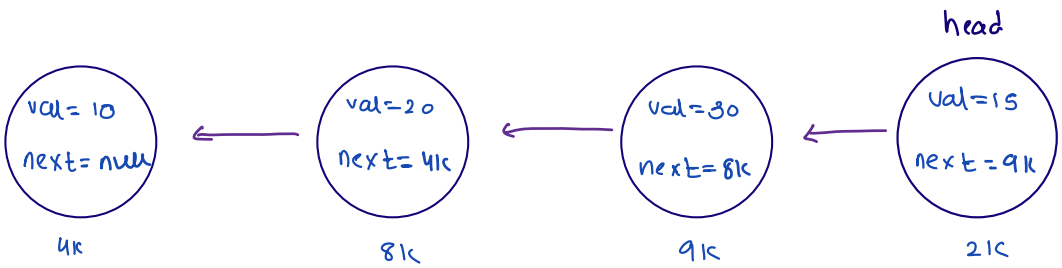
null

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

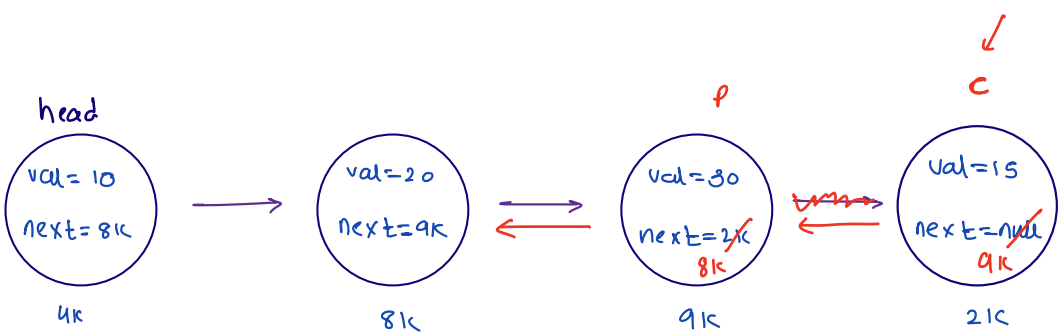
Q.4 Given head of LL, reverse it and return head of reversed LL.



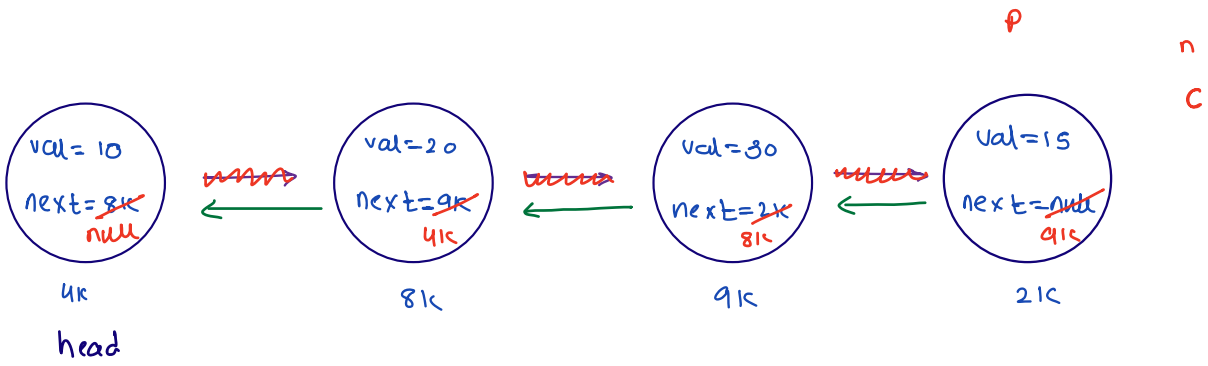
after reversal



hint



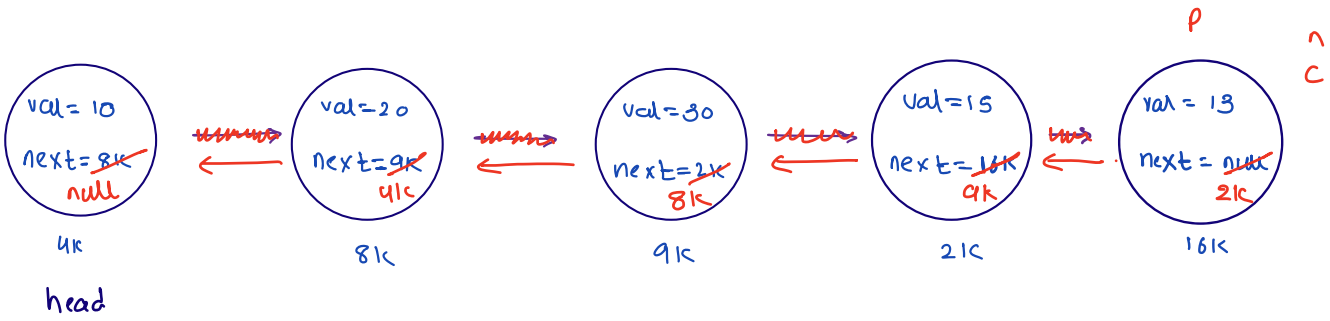
c.next = p




```
n = c.next;
c.next = p;
p = c;
c = n;
```

return p

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



return p => 16K

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

Doubts

1) group anagram

arr : [cat, bat, ate, tab, eat, toe, act, tea]

HM

key → HM (freq map)

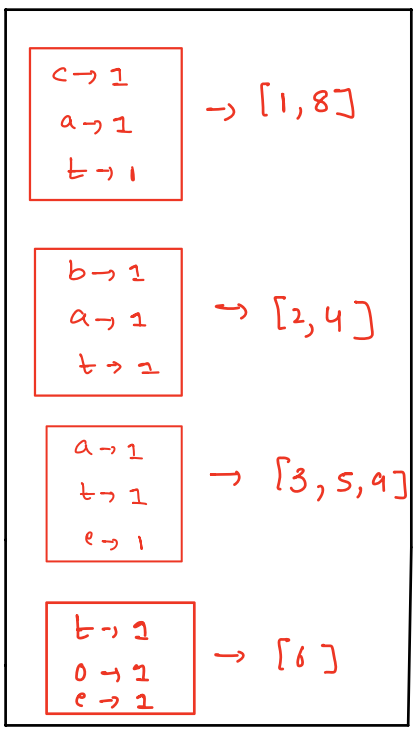
val → AL < Integers >

→ cat, act

→ bat, tab

→ ate, eat, tea

→ toe



Pairs II

target = 10

=

1 3 4 4 4 5 6 6 7 7 7 8 9

i

j

count = 1 + 3 + 6

window substring

A = ADOBECODEBANC

B = ABC

i

A D O B E C O D E B A N C

j

A → 1
B → 1
C → 1

Bmap

A → 1

mtc = ~~0 1 1 2 2 3 3 3~~ 2

ans = ~~A D O B E C~~

~~E B A N C~~

BANC

N → 1

C → 1

A = b a b a a b b g d c

i

j

B = b a b d

b → 2

g → 1

d → 1

mtc = ~~ø~~ ~~x~~ ~~x~~

~~ø~~ ~~x~~ 3

b → 2
a → 1
d → 1

B map

ans = ~~b a b a a b b g d~~

~~a b a a b b g d~~

~~b a a b b g d~~

~~a a b b g d~~

a b b g d