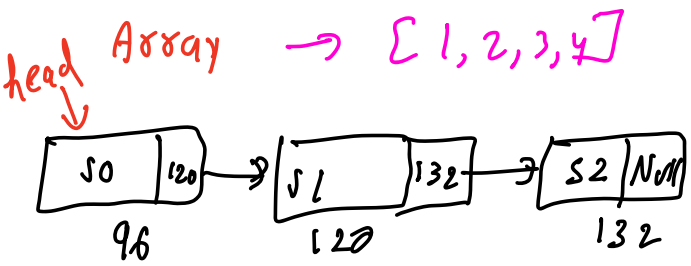
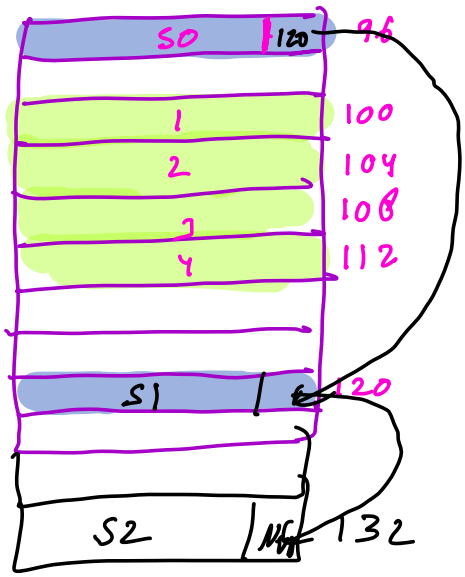
# Link List 1

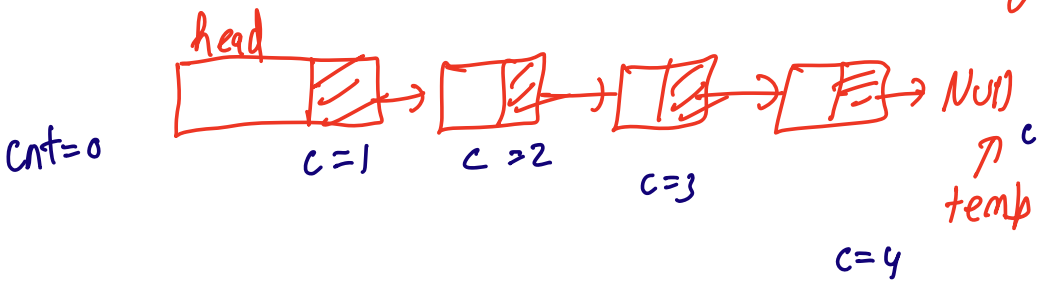



```
class Node
{
    int data
    Node next;
    public Node( int a)
    {
        this.data = a;
        this.next = null;
    }
}
```

temp = new Node(2);

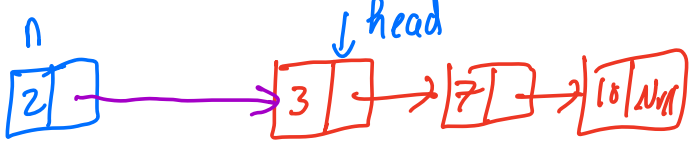

Q ⇒ Given a LL. Return the length!




```
int size (Node head)
{
    Node temp = head;
```

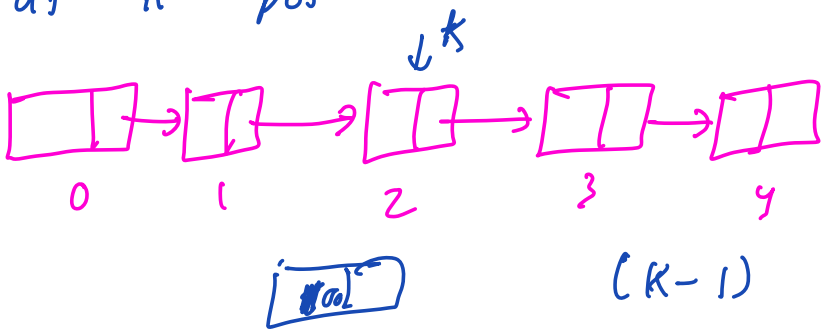
```
int count = 0
while (temp != null)
{
    count ++;
    temp = temp.next;
}
return count;
}
```

Insert values at start




```
Node n = new Node(2);
n.next = head
head = n;
```

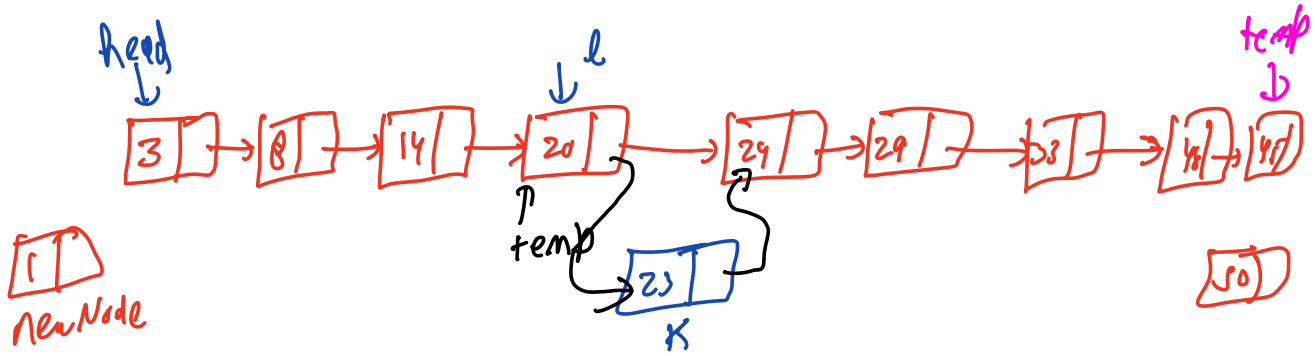
Adding at $k^{th}$ pos



Edge cases:

→ Head is null
→ LL has 1 / 2 / 3
→ Insert at start / end
→ Delete

Q⇒ Given a link list sorted in Asc order. Insert a value in correct position in sorted order




$K.next = l.next$

$l.next = K$

```
Node insertInSortedOrder ( Node head, K)
{
    Node newNode = new Node (K);

    if ( head == null)
        return newNode;
```

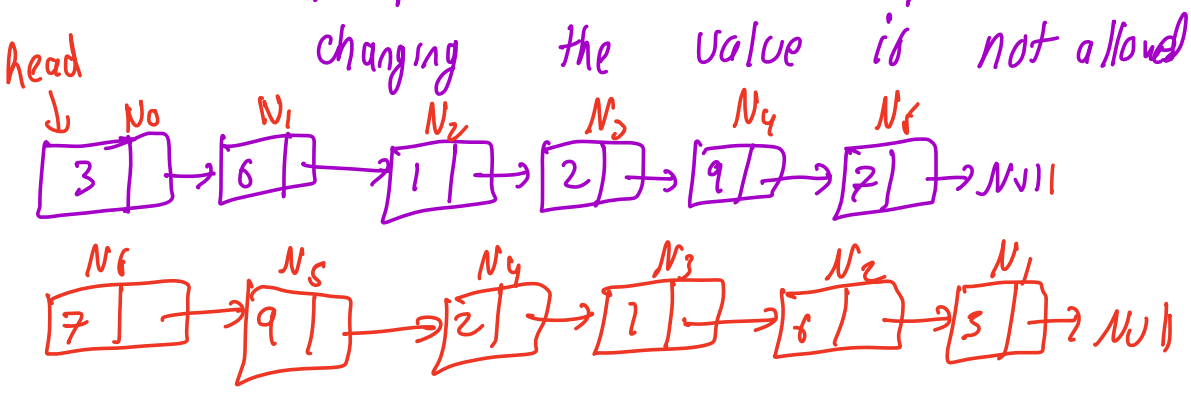
```
if (k <= head.value)
    newNode.next = head
    return newNode;

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

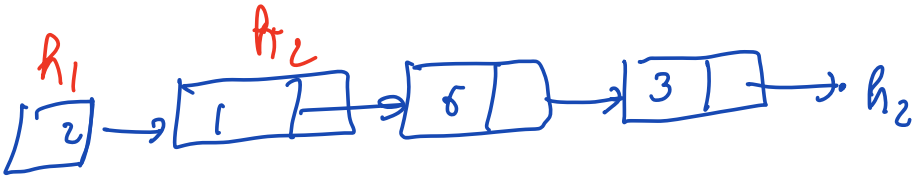
Q ⇒ Reverse the link list

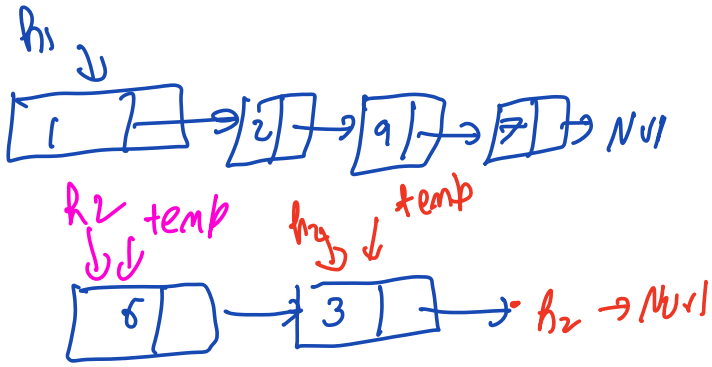
In-place : no extra space

changing the value is not allowed




R1

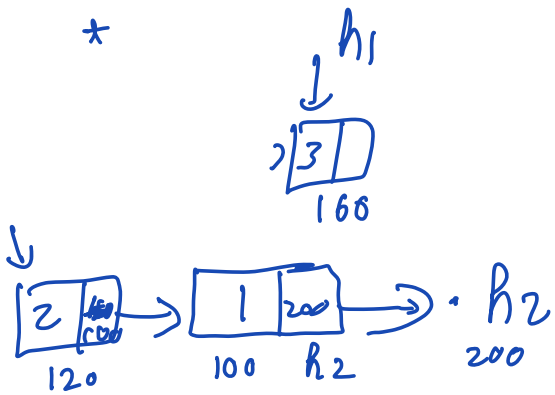



```
while (h1 != null)
{
    temp = h1;
    h1 = h1.next
    temp.next = h2
    h2 = temp;
}
```

```
Node reverse (Node Head)
{
    Node h2 = null, h1 = head, t = h1;
    while (h1 != null)
    {
        t = h1
        h1 = h1.next
        temp.next = h2
        h2 = temp;
    }
```

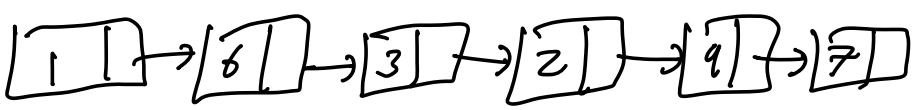
ret h2;

}




Q⇒ Given a LL. Reverse the first k nodes

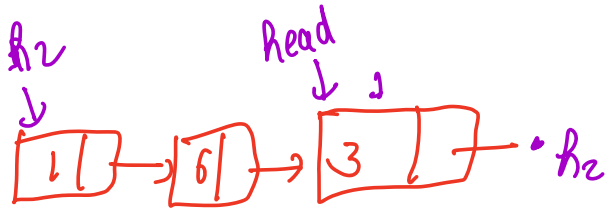

k = 3



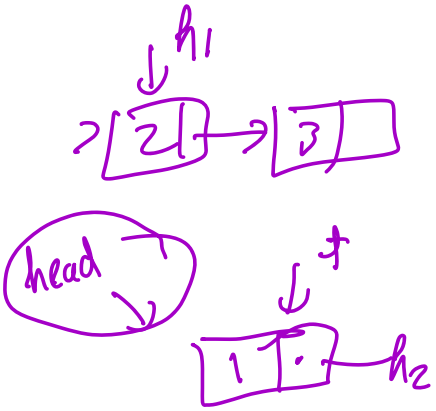

head = reverseFirstK( ⟶ )

```
Node    reverseFirstK( head, k)
{
    if ( head ==null || k==0 ) ret head;
  h2 = NULL    h1 = head;
  while ( h1 != null && k>0 )
    {
      t= h1;
```

```
        h1 = h1.next;
        t.next = h2
        h2 = t;
        k--;
    }
    head.next = h1;
        ret h2;
}
```


Break: 0 : 5 0

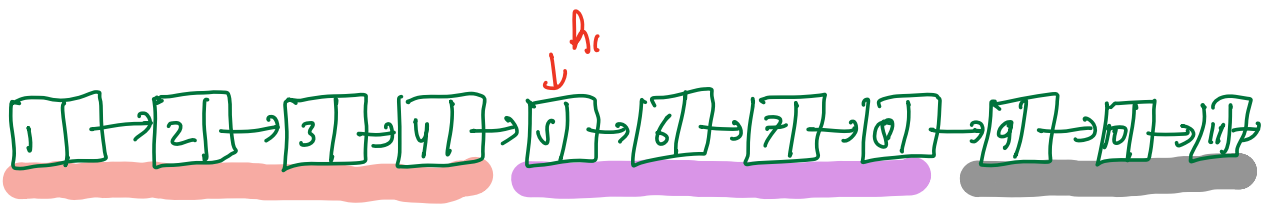


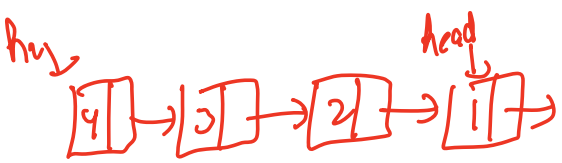
Google ⇒ Reverse in K groups

Given a LL. Reverse all sublist of size K.

K = 4






head.next = reverseInKGrp(h1, K)

```
Node reverseInKGroups(Node head, K)
{
    // Assumption: revInKgp(node, K) will
       return all groups of size K in list
       string from node & return new head

    ( K                 || head == null)
```

```
    |          return head;
h2 = NULL , h1 = head;

  count=k
while ( h1 != null && k>0 )
  {
    t = h1;
    h1 = h1.next;
    t.next = h2
    h2 = t;
     k--;
  }
  head.next = reverse In K Grp (h1, count);
   return h2;
}
```

TC: O(N)
SC: O(N)

K=2
N/2