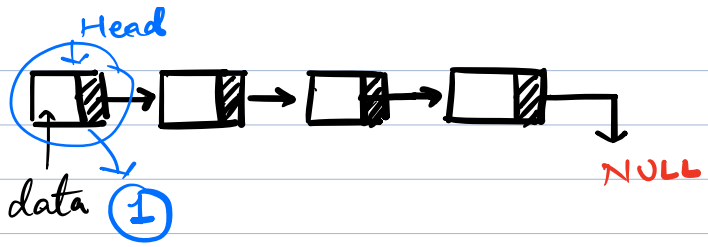
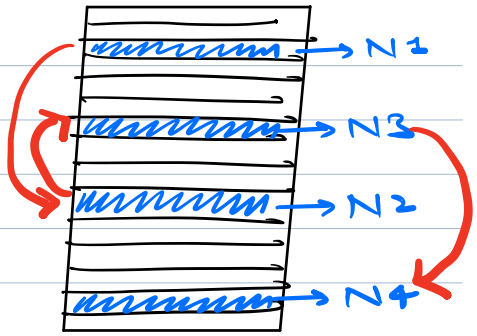




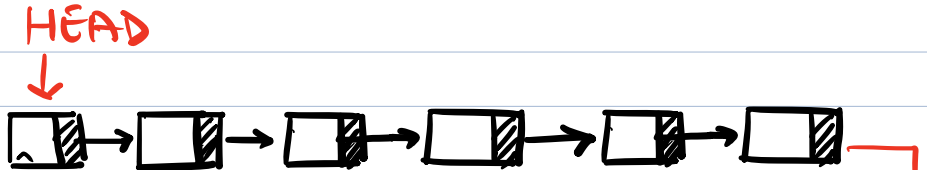

```
class Node {

    int data;
    Node next;

    public Node(int a) {
        this.data = a;
        this.next = NULL;
    }

}
```

Class constructor

```
Node node1 = new Node(5);
```

new → memory allocation

Given the head of a Linked list.
Return the length of the Linked list.



NULL
↑
temp

Node temp = HEAD;

Count = ~~1 2 3 4 5~~ 6

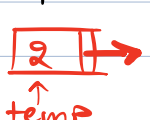
Code

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

Q Given an array of integers.
Create a LL of this array.

A: 2, [6, -5, 4, 1, 7]

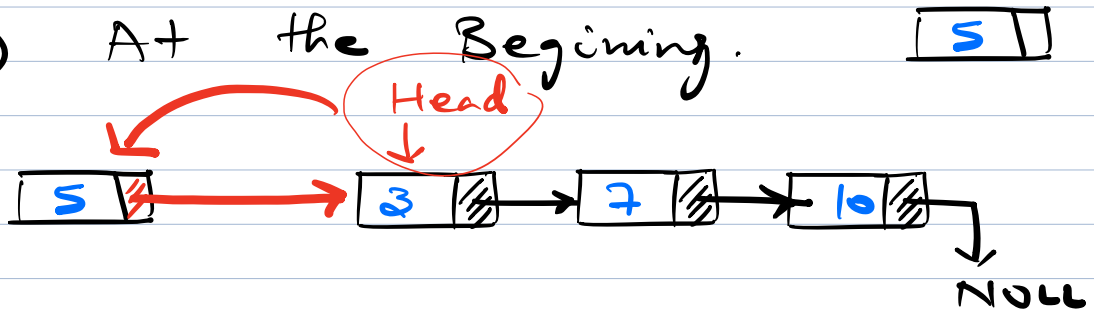
Soln



```
Node head = New Node (A[0]);
Node temp = head;
for (i = 1; i < N; i++) {
    temp.next = new Node (A[i]);
    temp = temp.next;
}
```

Insertion

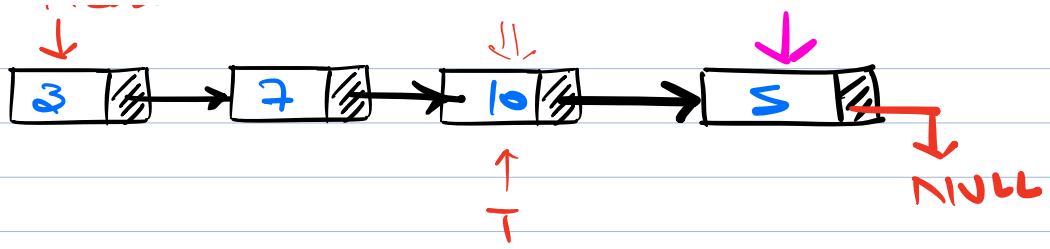
1) At the Beginning.




```
Node n = new Node (5);
n.next = Head;
Head = n;
```

2) At the End

Head




```
Node temp = head;
while ( temp.next != NULL) {
        temp = temp.next;
}
temp.next = new Node (5);
```

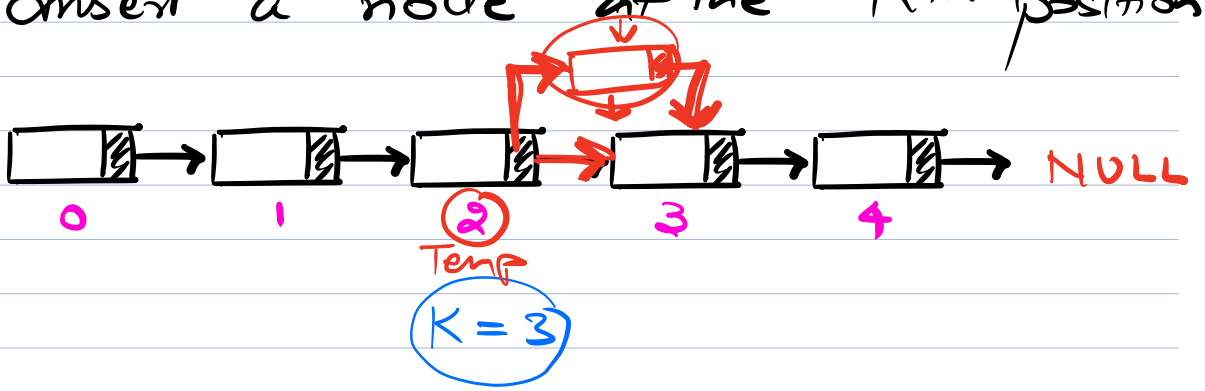
T.C. = O(N)

⇒ This can be optimised by using a tail pointer

↓

Last node of LL

3) Insert a node at the Kth position

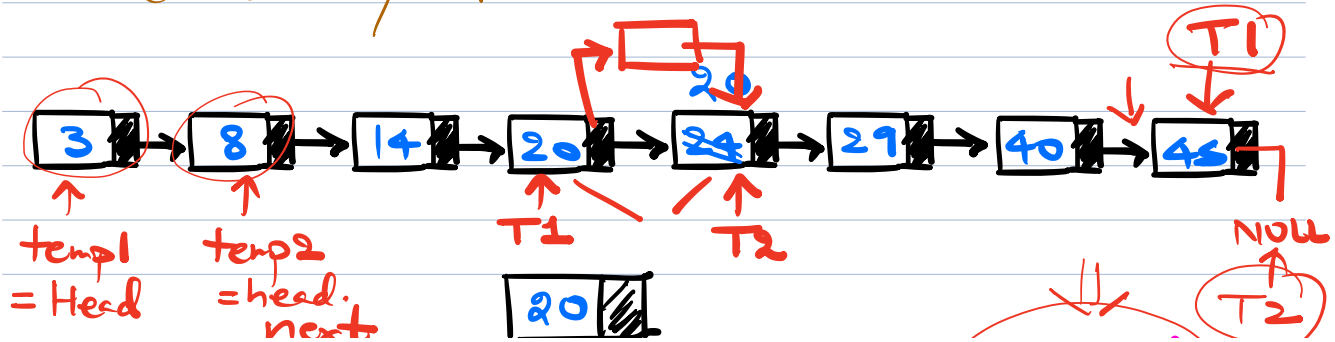


```
Node temp = head;
index = 0;
while ((index != K-1) && (temp.next!=NULL)
    index ++;
    temp = temp.next;
}
Node newNode = new Node(x);
newNode.next = temp.next;
temp.next = newNode;
```

T.C. = O(K)

Given the head node of a LL sorted in ASC order.

Insert a new given node on its correct position in the sorted LL



SO

1) 2 pointers

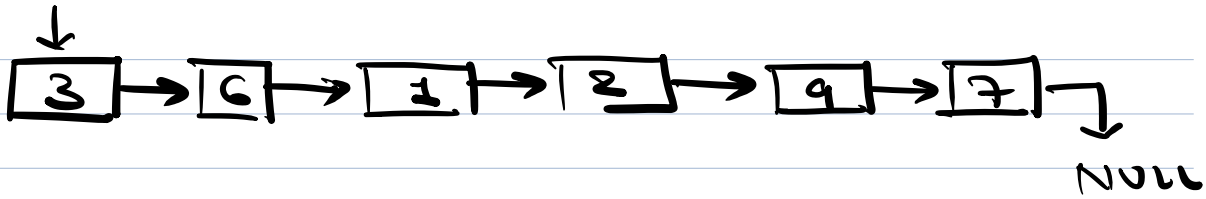
```
Node temp1 = head;
Node temp2 = head.next;

while( temp2 != NULL) {
    if (newNode.data > temp1.data
        &&
        newNode.data <= temp2.data){
            break;
    }

    temp1 = temp1.next;
    temp2 = temp2.next;
}

newNode.next = temp2;
temp1.next = newNode;
```

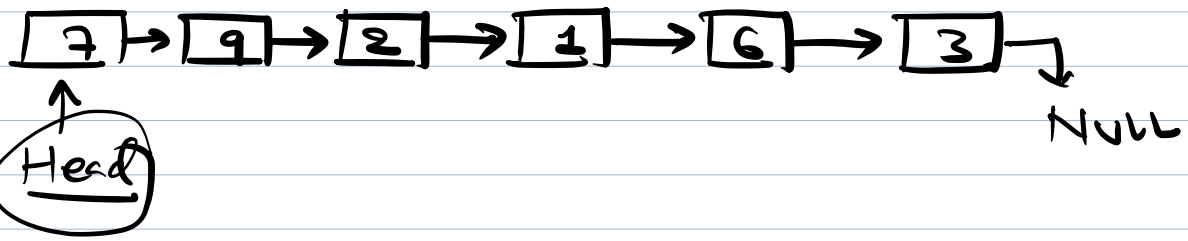
Reverse the LL

↳ Inplace → w/o using any extra space

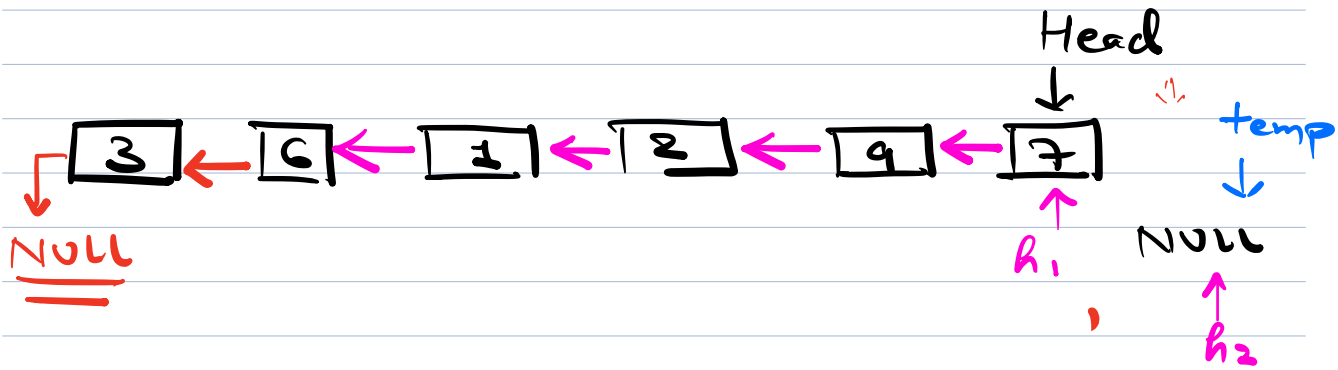
Changing the data of a node is not allowed.

Head





$h_1.next = h_2$



Node h_1 = head
Node h_2 = head.next

while ($h_2 != NULL$) {

temp = h_2.next ;

h_2.next = h_1 ;

$h_1 = h_2$

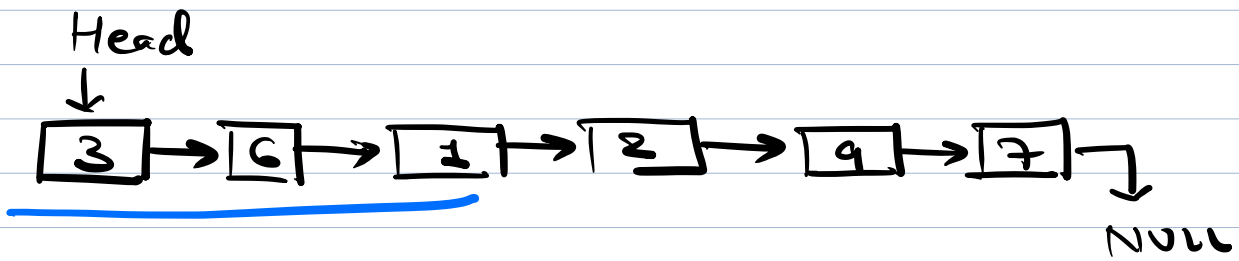
h_2 = temp

```
}
    Head.next = NULL
    Head = h1;

return HEAD;
```

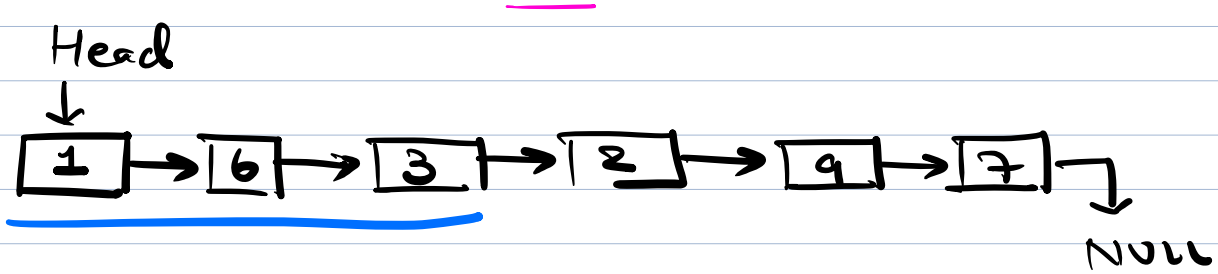
T.C. = O(N)

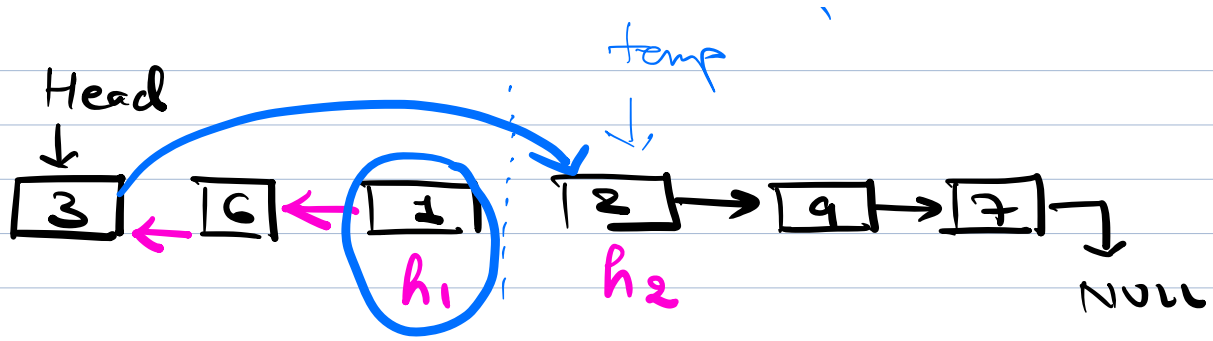
Q Given a LL

Reverse the first K nodes.



K = 3





count = 2

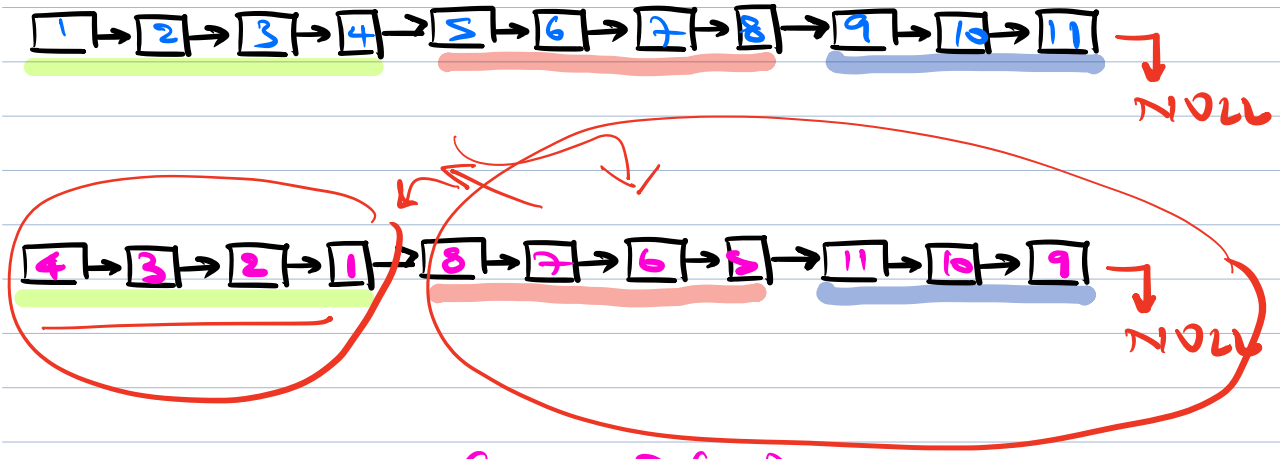
K Nodes ⇒ (K-1) links

```
Node h1= head;
Node h2 = head.next

count = 0;

while (count < (K-1)) {

    temp = h2.next;

    h2.next = h1;

    h1 = h2
    h2 = temp

}

head.next = H2;

Head = h1;
```

Q Reverse the LL in groups of K.

Google

K = 4



T.C. = O(N)

Doubt