- **HARD**
- **Max Score: 50 Points**
- https://course.acciojob.com/idle?question=3ef6a7a6-3c66-4508-9562-1ae709a7c501
-

Reverse In Groups Of K

Complete a function that takes a linked list and an integer value k as input and returns a new linked list with the nodes reversed in groups of k.

If the number of nodes is not divisible by k, the remaining nodes at the end should not be reversed.

Note

The values in the nodes should not be modified, only the order of the nodes should be changed.

Input Format

Input is handled by driver code you will be given the `head` of the linked list and the `k` value.

Output Format

Return the modified linked list's head.

Example 1

Input

```
5 2
1 2 3 4 5
```

Output

```
2 1 4 3 5
```

Explanation

```
In this case we reverse the groups (1,2), (3,4) and 5 is left as is.
```

Example 2

Input

```
8 3
1 2 3 4 5 6 7 8
```

Output

```
3 2 1 6 5 4 7 8
```

Explanation

```
In this case we reverse the groups (1,2,3), (4,5,6), the left nodes 7 and 8 are left as is.
```

Constraints

1 <= n,k<= 10^5

1 <= nodes.val <= 10^5

Topic Tags

- Recursion
- Linked lists

My code

```
import java.util.*;

class Node {
    int val;
    Node next;

    Node(int d) {
        val = d;
        next = null;
    }
}

class LinkedList {
    Node head, tail;

    void push(Node new_node) {
        if (head == null && tail == null) {
            head = tail = new_node;
            return;
        }
        tail.next = new_node;
        tail = new_node;
    }

    void printList(Node head) {
        Node temp = head;
        while (temp != null) {
            System.out.print(temp.val + " ");
            temp = temp.next;
        }
        System.out.println();
    }
}

class Solution {
    public Node reverseKGroup(Node head, int k) {
```

```
        //Write code here
                        if(head==null)
        return head;
      Stack<Node> st=new Stack<>();
       Node n=new Node(0);
      Node p=n;
       while( head!=null)
       {
           st.push(head);
            head=head.next;
           if(st.size() ==k)
           {
            while(!st.isEmpty())
            {
                p.next=st.pop();
                p=p.next;
                p.next=null;
            }
           }

       }
       //now if nod is not divisibal by k then add left node
       Node a=null;
        while(!st.isEmpty())//help to got stak at 1st position
           {
               a=st.pop();
           }
           p.next=a;
           return n.next;
    }
}

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int k = sc.nextInt();
        LinkedList llist = new LinkedList();
        int h1 = sc.nextInt();
        Node head = new Node(h1);
        llist.push(head);
        for (int i = 1; i < n; i++) {
            int data = sc.nextInt();
            llist.push(new Node(data));
```

```
        }
        Solution Obj = new Solution();
        head = Obj.reverseKGroup(head, k);
        llist.printList(head);
        sc.close();
    }
}
```