https://course.acciojob.com/idle?question=9391bb3a-b600-416a-a140-45647675ce52

- MEDIUM
- Max Score: 40 Points

Remove Nth Node From End of List

Given the head of a linked list, remove the nth node from the end of the list and return its head.

Note: You just need to complete removeNthFromEnd() function and return the new head of the linked list.

Input Format

The first line contains two spaced integers `k` and `n` where `k` denotes the length of `linked list` and `n` denotes the `nth` Node from the end . Next line contains `k` spaced integers representing the Nodes of the `List`.

Output Format

Print the new Linked List .

Example 1

Input

```
6 2
1 2 3 4 5 6
```

Output

```
1 2 3 4 6
```

Explanation

```
2nd Node from the end is Node with value 5

We remove it and update the List  : 1-> 2-> 3-> 4-> 6
```

Example 2

Input

```
5 4
7 6 5 4 3
```

Output

```
7 5 4 3
```

Explanation

```
4th Node from the end is Node with value 6

We remove it and update the list : 7-> 5-> 4-> 3
```

Constraints

1 <= k <= 30

1 <= Node.val < 100

1 <= n <= k

Topic Tags

- **Linked lists**

My code

```
// n java
import java.io.*;
import java.util.*;
class Node
    {
        int data;
        Node next;
        Node(int d) {data = d; next = null; }
    }
class insertion
{
    Node head;
    Node tail;
        public void addToTheLast(Node node)
        {
          if (head == null)
          {
           head = node;
           tail = node;
          }
          else
          {
           tail.next = node;
           tail = node;
          }
        }
      void printList(Node head)
    {
        Node temp = head;
        while (temp != null)
        {
           System.out.print(temp.data+" ");
           temp = temp.next;
        }
        System.out.println();
    }
        /* Drier program to test above functions */

}
class Main
{
    public static void main(String args[])throws IOException
    {
         BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));

            String S[] = br.readLine().split(" ");
                        int N = Integer.parseInt(S[0]);
            int n = Integer.parseInt(S[1]);


                    String S1[] = br.readLine().split(" ");
                        insertion llist = new insertion();
                        int a1=Integer.parseInt(S1[0]);
                        Node head= new Node(a1);
            llist.addToTheLast(head);
            for (int i = 1; i < N; i++)
                        {
                                int a = Integer.parseInt(S1[i]);
                                llist.addToTheLast(new Node(a));
                        }

          Solution ob = new Solution();
                    Node newhead=ob.removeNthFromEnd(llist.head, n);
                    llist.printList(newhead);
     }
}

class Solution
{
     public static Node removeNthFromEnd(Node head, int n) {
     //Write your code here
                    Node r=head;
                    int c=0;
                    while(r!=null)
                         {
                                   r=r.next;
                                   c++;
                         }
                    n=c-n+1;
                    r=head;
                    if(n==1)
                         return r.next;
                    for(int i=1;i<n-1;i++)
                         r=r.next;
                    //now remove
                    r.next=r.next.next;

                    return head;
```

```
}
}
```