**Delete node in Doubly Linked List**

Given a doubly linked list and a position. The task is to delete a node from given position in a doubly linked list.

**Example 1:**

**Input:**

LinkedList = 1 <--> 3 <--> 4

x = 3

**Output:** 1 3

**Explanation:** After deleting the node at

position 3 (position starts from 1),

the linked list will be now as 1->3.

**Example 2:**

**Input:**

LinkedList = 1 <--> 5 <--> 2 <--> 9

x = 1

**Output:** 5 2 9

**Expected Time Complexity** : O(N)  
**Expected Auxilliary Space** : O(1)

**Constraints:**  
2 <= size of the linked list <= 1000  
1 <= x <= N

**Company Tags**

[**Amazon**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Amazon) [**Walmart**](https://practice.geeksforgeeks.org/explore/?company%5b%5d=Walmart)

//{ Driver Code Starts

//Initial Template for Java

import java.util.\*;

class Node

{

int data;

Node next;

Node prev;

Node(int d)

{

data = d;

next = prev = null;

}

}

class Delete\_Node\_From\_DLL

{

Node head;

Node tail;

void addToTheLast(Node node)

{

if(head == null)

{

head = node;

tail = node;

}

else

{

tail.next = node;

tail.next.prev = tail;

tail = tail.next;

}

}

void printList(Node head)

{ Node temp = head;

while(temp != null)

{

System.out.print(temp.data+" ");

temp = temp.next;

}

System.out.println();

}

public static void main(String args[])

{

Scanner sc = new Scanner(System.in);

int t = sc.nextInt();

while(t>0)

{

int n = sc.nextInt();

Delete\_Node\_From\_DLL list = new Delete\_Node\_From\_DLL();

int a1 = sc.nextInt();

Node head = new Node(a1);

list.addToTheLast(head);

for(int i=1;i<n;i++)

{

int a = sc.nextInt();

list.addToTheLast(new Node(a));

}

a1 = sc.nextInt();

Solution g = new Solution();

//System.out.println(list.temp.data);

Node ptr = g.deleteNode(list.head, a1);

list.printList(ptr);

t--;

}

}

}

// } Driver Code Ends

class Solution

{

// function returns the head of the linkedlist

Node deleteNode(Node head,int x)

{

Node curr=head;

if(x==1){

head=head.next;

head.prev=null;

return head;

}

Node pre=null;

while(curr!=null && x>1){

x--;

pre=curr;

curr=curr.next;

}

if(curr.next!=null){

curr.next.prev=pre;

}

pre.next=curr.next;

return head;

}

}