**Problem Name:** Delete node in a linked list

**Topics:** Linked list

**Companies:** Facebook, Microsoft, Amazon, Google, Apple, Bloomberg, Qualcomm, Yahoo, Cisco

**Level:** Easy

**Language:** C++

**Problem Statement**: Write a function to **delete a node** in a singly-linked list. You will **not** be given access to the head of the list, instead you will be given access to **the node to be deleted** directly.

It is **guaranteed** that the node to be deleted is **not a tail node** in the list.

**Input Format:**

First line of the input contain integer n (size of list)

Second line contain n space separated integer list values.

Last line contain integer value pos representing value of node to delete.

Ex:

5

1 2 3 4 5

1

**Output Format:** Print linked list after removing node having value pos

**Constraints:**

* The number of the nodes in the given list is in the range [2, 1000].
* -1000 <= Node.val <= 1000
* The value of each node in the list is **unique**.
* The node to be deleted is **in the list** and is **not a tail** node

**Examples:**

**Input:** head = [4,5,1,9], node = 5

**Output:** [4,1,9]

**Explanation:** You are given the second node with value 5, the linked list should become 4 -> 1 -> 9 after calling your function.

**Solution:**

**Explanation:**

Consider an example : 1->2->3->4->5

1. We are supposed to delete say **2** i.e. **node points to 2** initially
2. We will do the following steps until **current->next->next!=NULL**  
   2a. **current->val = current->next->val**  
   2b. **current = current->next**
3. By doing this we will get  
   1->2->3->4->5  
   ...... \*  
   1->3->3->4->5  
   ...............\*  
   1->3->4->4->5 (**current->next->next is NULL so break**)  
   .......................\*
4. Finally we will do the following 2 steps  
   4a. **current->data = current->next->data**  
   4b. **current->next = NULL**  
   1->3->4->5->5  
   ......................\*  
   1->3->4->5->X  
   .....................\*

**Code:**

#include <bits/stdc++.h>

using namespace std;

class ListNode

{

    public:

        int val;

        ListNode\* next;

        ListNode(int a){

            val = a;

            next = NULL;

        }

};

void insertNode(ListNode\* &head,int val) {

    ListNode\* newNode = new ListNode(val);

    if(head == NULL) {

        head = newNode;

        return;

    }

    ListNode\* temp = head;

    while(temp->next != NULL)

     temp = temp->next;

    temp->next = newNode;

    return;

}

void printList(ListNode \*node)

{

    while (node!=NULL)

    {

        cout<<node->val<<" ";

        node = node->next;

    }

}

void deleteNode(ListNode\* node) {

    ListNode\* current = node;

    while(current->next->next!=nullptr){

        current->val = current->next->val;

        current = current->next;

    }

    current->val = current->next->val;

    current->next = nullptr;

}

int main()

{

    ListNode\* a = NULL;

    ListNode\* del = NULL;

    int n, temp, pos;

    cin>>n;

    while(n--){

        cin>>temp;

        insertNode(a, temp);

    }

    del = a;

    cin>>pos;

    while(del !=NULL && del->val != pos){

        del = del->next;

    }

    if(del)

        deleteNode(del);

    printList(a);

    return 0;

}

**Time Complexity**: O(N)

**Space Complexity:** O(1)

**Optimized Solution:**

**Explanation:**

Basically we are just copying node's next value to itself  
Input: 1->2->3->4->5 ->6  
given node = 4  
output: 1->2->3->5->6

**1.** in first step: node->val becomes node's next ki value i.e 4 becomes 5  
1->2->3->5->5->6  
**2.** in 2nd step: node points to node's next next i.e 5->6  
1->2->3->5->6

**Code:**

#include <bits/stdc++.h>

using namespace std;

class ListNode

{

    public:

        int val;

        ListNode\* next;

        ListNode(int a){

            val = a;

            next = NULL;

        }

};

void insertNode(ListNode\* &head,int val) {

    ListNode\* newNode = new ListNode(val);

    if(head == NULL) {

        head = newNode;

        return;

    }

    ListNode\* temp = head;

    while(temp->next != NULL)

     temp = temp->next;

    temp->next = newNode;

    return;

}

void printList(ListNode \*node)

{

    while (node!=NULL)

    {

        cout<<node->val<<" ";

        node = node->next;

    }

}

void deleteNode(ListNode\* node) {

    node->val = node->next->val;

    node->next = node->next->next;

}

int main()

{

    ListNode\* a = NULL;

    ListNode\* del = NULL;

    int n, temp, pos;

    cin>>n;

    while(n--){

        cin>>temp;

        insertNode(a, temp);

    }

    del = a;

    cin>>pos;

    while(del !=NULL && del->val != pos){

        del = del->next;

    }

    if(del)

        deleteNode(del);

    printList(a);

    return 0;

}

**Time Complexity**: O(1)

**Space Complexity:** O(1)