Given only a pointer to a node to be deleted in a singly linked list, how do you delete it?

A **simple solution** is to traverse the linked list until you find the node you want to delete. But this solution requires pointer to the head node which contradicts the problem statement.

Fast solution is to copy the data from the next node to the node to be deleted and delete the next node. Something like following.

```
    struct node *temp  = node_ptr->next;
    node_ptr->data  = temp->data;
    node_ptr->next  = temp->next;
    free(temp);
```

Program:

C

```
#include<stdio.h>
#include<assert.h>
#include<stdlib.h>

/* Link list node */
struct node
{
    int data;
    struct node* next;
};

/* Given a reference (pointer to pointer) to the head
of a list and an int, push a new node on the front
of the list. */
void push(struct node** head_ref, int new_data)
{
    /* allocate node */
   struct node* new_node =
             (struct node*) malloc(sizeof(struct node));

   /* put in the data  */
   new_node->data  = new_data;

   /* link the old list off the new node */
   new_node->next = (*head_ref);

   /* move the head to point to the new node */
   (*head_ref)    = new_node;
}

void printList(struct node *head)
{
   struct node *temp = head;
   while(temp != NULL)
   {
      printf("%d  ", temp->data);
      temp = temp->next;
   }
}

void deleteNode(struct node *node_ptr)
{
   struct node *temp = node_ptr->next;
   node_ptr->data    = temp->data;
   node_ptr->next    = temp->next;
   free(temp);
}
```

```
}

/* Drier program to test above function*/
int main()
{
    /* Start with the empty list */
    struct node* head = NULL;

    /* Use push() to construct below list
    1->12->1->4->1  */
    push(&head, 1);
    push(&head, 4);
    push(&head, 1);
    push(&head, 12);
    push(&head, 1);

    printf("\n Before deleting \n");
    printList(head);

    /* I m deleting the head itself.
        You can check for more cases */
   deleteNode(head);

   printf("\n After deleting \n");
   printList(head);
   getchar();
}
```

Java

```
// Java program to del the node in which only a single pointer
// is known pointing to that node

class LinkedList {

    static Node head;

    static class Node {

        int data;
        Node next;

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

    void printList(Node node) {
        while (node != null) {
            System.out.print(node.data + " ");
            node = node.next;
        }
    }

    void deleteNode(Node node) {
        Node temp = node.next;
        node.data = temp.data;
        node.next = temp.next;
        System.gc();

    }

    // Driver program to test above functions
    public static void main(String[] args) {
        LinkedList list = new LinkedList();
        list.head = new Node(1);
        list.head.next = new Node(12);
        list.head.next.next = new Node(1);
        list.head.next.next.next = new Node(4);
        list.head.next.next.next.next = new Node(1);

        System.out.println("Before Deleting ");
        list.printList(head);

        /* I m deleting the head itself.
         You can check for more cases */
        list.deleteNode(head);
        System.out.println("");
        System.out.println("After deleting ");
        list.printList(head);
    }
}

// This code has been contributed by Mayank Jaiswal
```

This solution doesn't work if the node to be deleted is the last node of the list. To make this solution work we can mark the end node as a dummy node. But the programs/functions that are using this function should also be modified.

Try this problem for doubly linked list.