Given only a pointer/reference 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);
   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("Before deleting \n");
    printList(head);

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

    printf("\nAfter deleting \n");
    printList(head);
    getchar();
    return 0;
}
```

Java

```
class LinkedList
{
     Node head; // head of the list

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

     /* Given a reference to the head of a list and an int,
        inserts a new Node on the front of the list. */
     public void push(int new_data)
     {
       /* 1. alloc the Node and put the data */
        Node new_Node = new Node(new_data);

        /* 2. Make next of new Node as head */
        new_Node.next = head;

        /* 3. Move the head to point to new Node */
        head = new_Node;
     }

     /* This function prints contents of linked list
        starting from the given Node */
     public  void printList()
     {
        Node tNode = head;
        while (tNode != null) {
           System.out.print(tNode.data+" ");
           tNode = tNode.next;
        }
     }

     public  void deleteNode(Node Node_ptr)
     {
        Node temp = Node_ptr.next;
        Node_ptr.data = temp.data;
        Node_ptr.next = temp.next;
        temp = null;
     }

     public static void main(String[] args)
     {
        LinkedList llist = new LinkedList();

        /* Use push() to construct below list
        1->12->1->4->1  */
        llist.push(1);
        llist.push(4);
        llist.push(1);
        llist.push(12);
        llist.push(1);

        System.out.println("Before deleting");
        llist.printList();

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

        System.out.println("\nAfter Deleting");
        llist.printList();
     }
}
// This code is contributed by Rajat Mishra
```

Output:

```
Before deleting
1 12 1 4 1
After deleting
12 1 4 1
```

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.

Exercise: Try this problem for doubly linked list.