Delete a given node in Linked List under given constraints

Given a Singly Linked List, write a function to delete a given node. Your function must follow following constraints:

1) It must accept pointer to the start node as first parameter and node to be deleted as second parameter i.e., pointer to head node is not global.

2) It should not return pointer to the head node.

3) It should not accept pointer to pointer to head node.

You may assume that the Linked List never becomes empty.

Let the function name be deleteNode(). In a straightforward implementation, the function needs to modify head pointer when the node to be deleted is first node. As discussed in previous post, when a function modifies the head pointer, the function must use one of the given approaches, we can't use any of those approaches here.

Solution

We explicitly handle the case when node to be deleted is first node, we copy the data of next node to head and delete the next node. The cases when deleted node is not the head node can be handled normally by finding the previous node and changing next of previous node. Following is C implementation.

C

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

/* structure of a linked list node */
struct node
{
    int data;
    struct node *next;
};

void deleteNode(struct node *head, struct node *n)
{
    // When node to be deleted is head node
    if(head == n)
    {
        if(head->next == NULL)
```

```
        {
            printf("There is only one node. The list can't be made empty ");
            return;
        }

        /* Copy the data of next node to head */
        head->data = head->next->data;

        // store address of next node
        n = head->next;

        // Remove the link of next node
        head->next = head->next->next;

        // free memory
        free(n);

        return;
    }


    // When not first node, follow the normal deletion process

    // find the previous node
    struct node *prev = head;
    while(prev->next != NULL && prev->next != n)
        prev = prev->next;

    // Check if node really exists in Linked List
    if(prev->next == NULL)
    {
        printf("\n Given node is not present in Linked List");
        return;
    }

    // Remove node from Linked List
    prev->next = prev->next->next;

    // Free memory
    free(n);

    return;
}

/* Utility function to insert a node at the begining */
void push(struct node **head_ref, int new_data)
{
    struct node *new_node =
        (struct node *)malloc(sizeof(struct node));
    new_node->data = new_data;
    new_node->next = *head_ref;
    *head_ref = new_node;
}

/* Utility function to print a linked list */
void printList(struct node *head)
{
    while(head!=NULL)
    {
        printf("%d ",head->data);
        head=head->next;
    }
    printf("\n");
}

/* Driver program to test above functions */
int main()
{
    struct node *head = NULL;
```

```
    /* Create following linked list
      12->15->10->11->5->6->2->3 */
    push(&head,3);
    push(&head,2);
    push(&head,6);
    push(&head,5);
    push(&head,11);
    push(&head,10);
    push(&head,15);
    push(&head,12);

    printf("Given Linked List: ");
    printList(head);

    /* Let us delete the node with value 10 */
    printf("\nDeleting node %d: ", head->next->next->data);
    deleteNode(head, head->next->next);

    printf("\nModified Linked List: ");
    printList(head);

    /* Let us delete the the first node */
    printf("\nDeleting first node ");
    deleteNode(head, head);

    printf("\nModified Linked List: ");
    printList(head);

    getchar();
    return 0;
}
```

Run on IDE

Java

```
// Java program to delete a given node in linked list under given constraints

class LinkedList {

    static Node head;

    static class Node {

        int data;
        Node next;

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

    void deleteNode(Node node, Node n) {

        // When node to be deleted is head node
        if (node == n) {
            if (node.next == null) {
                System.out.println("There is only one node. The list "
                                   + "can't be made empty ");
                return;
            }

            /* Copy the data of next node to head */
            node.data = node.next.data;

            // store address of next node
```

```
            n = node.next;

            // Remove the link of next node
            node.next = node.next.next;

            // free memory
            System.gc();

            return;
        }

        // When not first node, follow the normal deletion process
        // find the previous node
        Node prev = node;
        while (prev.next != null && prev.next != n) {
            prev = prev.next;
        }

        // Check if node really exists in Linked List
        if (prev.next == null) {
            System.out.println("Given node is not present in Linked List");
            return;
        }

        // Remove node from Linked List
        prev.next = prev.next.next;

        // Free memory
        System.gc();

        return;
    }

    /* Utility function to print a linked list */
    void printList(Node head) {
        while (head != null) {
            System.out.print(head.data + " ");
            head = head.next;
        }
        System.out.println("");
    }

    public static void main(String[] args) {
        LinkedList list = new LinkedList();
        list.head = new Node(12);
        list.head.next = new Node(15);
        list.head.next.next = new Node(10);
        list.head.next.next.next = new Node(11);
        list.head.next.next.next.next = new Node(5);
        list.head.next.next.next.next.next = new Node(6);
        list.head.next.next.next.next.next.next = new Node(2);
        list.head.next.next.next.next.next.next.next = new Node(3);

        System.out.println("Given Linked List :");
        list.printList(head);
        System.out.println("");

        // Let us delete the node with value 10
        System.out.println("Deleting node :" + head.next.next.data);
        list.deleteNode(head, head.next.next);

        System.out.println("Modified Linked list :");
        list.printList(head);
        System.out.println("");

        // Lets delete the first node
        System.out.println("Deleting first Node");
        list.deleteNode(head, head);
        System.out.println("Modified Linked List");
        list.printList(head);
```

```
    }
}
// this code has been contributed by Mayank Jaiswal
```

Run on IDE

Output:

```
Given Linked List: 12 15 10 11 5 6 2 3

Deleting node 10:
Modified Linked List: 12 15 11 5 6 2 3

Deleting first node
Modified Linked List: 15 11 5 6 2 3
```

Please write comments if you find the above codes/algorithms incorrect, or find other ways to solve the same problem.

Company Wise Coding Practice Topic Wise Coding Practice

43 Comments Category: Linked Lists

Related Posts:

- Convert a Binary Tree to a Circular Doubly Link List
- Subtract Two Numbers represented as Linked Lists
- Rearrange a given list such that it consists of alternating minimum maximum elements
- Flatten a multi-level linked list | Set 2 (Depth wise)
- Decimal Equivalent of Binary Linked List

- Merge K sorted linked lists
- Check if a linked list is Circular Linked List
- Delete middle of linked list

(Login to Rate and Mark)

1.7 Average Difficulty : **1.7/5.0**
Based on **48** vote(s)

Add to TODO List
Mark as DONE

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.

43 Comments **GeeksforGeeks** 1 **Login**

Recommend 3 **Share** **Sort by Newest**

Join the discussion…

Neel Shah • a month ago

my deleteNode function....

```
void deleteNode(struct node *head, struct node *n)
{
struct node * curr = head;
while(curr != n && curr != NULL)
curr = curr-> next;

if(head->next == NULL)
{
printf("There is only one node. The list can't be made empty ");
return;
}

if(curr == NULL)
{
printf("element is not here\n");
return ;
}
```

see more

• Reply • Share ›

Raghav M • 3 months ago

i having strings like
first string "raghava"
second string "rama"

compare both strings of first two letters and print the remaining all charecters of first string

please send me the logic of above discussion

Reply • Share ›

Chaitanya Reddy ➚ Raghav M • 3 months ago

though question is quite not informative i assumed that if first two characters of two strings are equal than return remaining characters in first string if not return chars in second string. I am sharing the JAVA logic using java's String class.

```
public static String compareStrings(String str,String str1){
return str.substring(0,2).equals(str1.substring(0,2))?
str.substring(2):str1.substring(2);
}
```

Reply • Share ›

Raghav M • 3 months ago

i having an n number of nodes delete last n-2 node in linked list and also traversing please send me the programme logic

\\\\\\\\\\\\\\\\\\\\\\

Reply • Share ›

Chaitanya Reddy ➚ Raghav M • 3 months ago

For traversing the list and print the nodes data

```
public void displayNodes(){
Node temp=head;
while(temp!=null){
System.out.println(temp.data);
temp = temp.next;
}
}
```

where you should have 'head' as class variable.
and for delete just go through the main geeksforgeeks thread code

Reply • Share ›

raghava meruguboyina ➚ Chaitanya Reddy • 3 months ago

For example:1->2->3->4->5->6

Traversing then head goes to 6th node and also last node then delete 4th node please send me the logic

Reply • Share ›

sam • 3 months ago

Suppose the list is : 1->2->3->4->5 (Node containing 5
points to NULL) . And the node to be deleted is the last one. Then after
deletion, node containing 4 should point to NULL. Your program does not
do this in such a case.
As a result, next time the loop will run infinite times since no node will point to NULL.

Reply • Share ›

Ritish Verma ➚ sam • 3 months ago

Nope it works since in the last line "prev->next = prev->next->next;" , since prev in your case is 4 and node to be deleted is 5 therefore "4 -> next = 5 -> next ".
5's next is Null therefore 4 points to null after 5 is deleted. Therefore no infinite loop occurs in the print function later.

1 Reply • Share ›

dhanraj • 3 months ago

plz help....i want program for delete node linked list without knowing head....plz fast

Reply • Share ›

Chaitanya Reddy ➚ dhanraj • 3 months ago

http://stackoverflow.com/quest...

Reply • Share ›

anonymo • 4 months ago

empty return?? Ouch. Its the worst you can do in programming. Its like you jump out the window.

2 Reply • Share ›

Pankaj Dabade • 9 months ago

```
void deleteNode(node* head, node* target) {
if (!head || !head->next) return;
if (head == target) {
target = head->next;
head->data = target->data;
}
while (head->next && head->next != target)
head = head->next;
if (head->next == target) {
head->next = target->next;
delete target;
}
return;
}
```

Reply • Share ›

rasen58 • 9 months ago

In void push(struct node **head_ref, int new_data), why do we pass a pointer to a pointer, instead of just the pointer to the head?

Reply • Share ›

Deepak Dodeja ➚ rasen58 • 9 months ago

Here we passed a reference of the head of linked list so to dereference we need two pointers.Remember in c language we use "Call by reference " to reflect the changes to the calling function,Similarly here changes get reflected by passing the refernce to the head of linked list and for that we need pointer to a pointer.

1 Reply • Share ›

rasen58 ➚ Deepak Dodeja • 9 months ago

But then why didn't we have to use two pointers for deleteNode?

Reply • Share ›

Deepak Dodeja ➚ rasen58 • 9 months ago

Yup, The same query i have, you can see my post below i also posted same question but not get reply yet :(

Reply • Share ›

Deepak Dodeja • 9 months ago

I am unable to understand how this is happening, since these all changes in linked list are local to deleteNode(), so how they can reflect back to the calling function? If reflects back then what is the need of "third method" of "http://www.geeksforgeeks.org/h..." , Please help...

Reply • Share ›

Rajan Kalra • a year ago

Was it needed to have the argument 'n' to be of type 'node'!

Reply • Share ›

AlexPere • a year ago

Why is your head a tail?

Reply • Share ›

Vaishali Singh • a year ago

your logic is wrong

Reply • Share ›

Sagar ➚ Vaishali Singh • a year ago

Agreed.

Reply • Share ›

rads • a year ago

Please correct me if I am wrong....I think ques asked is something else and ans is something else....according to me question says we are given 2 linked lists and the nodes which are common in the 2 lists should be removed .Can anyone please explain me how the given solution is the implementation of the question
Thank you

^ | v • Reply • Share ›

Nikhil Jindal • 2 years ago

cant we use the pointer to pointer to pointer to head ie ***head_ref?

^ | v • Reply • Share ›

SlickHackz ➚ Nikhil Jindal • a year ago

No.. See Constraint 3 (It should not accept pointer to pointer to head node.)

1 ^ | v • Reply • Share ›

anonymous • 2 years ago

how will you delete the last remaining element in list?

1 ^ | v • Reply • Share ›

danny ➚ anonymous • 2 years ago

U can't remove the last element in list, the question also specifies that the list should not be emptied...

2 ^ | v • Reply • Share ›

Guest ➚ anonymous • 2 years ago

find the prev node i.e the second last node and make it'e next to NULL...

3 ^ | v • Reply • Share ›

GeeksITGeeks • 2 years ago

I am getting segmentation fault on line :
while(n != count) on below code, if any idea please help....

```
void deleteNthNode()
{
struct node *current = start;
struct node *new, *first;
int n, num = 0;
printf ("\nEnter the node to delete : ");
scanf ("%d", &n);
if (current -> next == NULL){
printf("\nThe single node can not be deleted");
return;
}
if (n == 1){
```

```
if (n == 1){
current -> data = current -> next -> data;
new = current -> next;
current -> next = current -> next -> next;
```

see more

• Reply • Share ›

yugu ➚ GeeksITGeeks • 2 years ago

segmentation fault because of the data type you used is incorrect and it is different

• Reply • Share ›

kavi • 3 years ago

I understand that if the list contains only a SINGLE NODE, and we wish to delete it, it CANNOT BE DONE using this program.

I can already see that statement , LIS "The list can't be made empty" , But still wish to confirm, that IT IS IMPOSSIBLE if the list contains single node?

Am I right? Please let me know if any other way if that single node can be deleted with the given constraint.
Thanks.

• Reply • Share ›

ashatm • 3 years ago

when the first node is to be deleted, you have copied the data of the second node into the first one and deleted the second node, but that actually does not delete the given node; also the constraint is not to return a pointer to head, how does that mean we can't update head in the delete function(in case we delete the first node only and update head to the second node)?

• Reply • Share ›

ashatm ➚ ashatm • 3 years ago

i get it... sorry for the silly doubt..

• Reply • Share ›

Nidhi ➚ ashatm • 3 years ago

How will the changes made in head pointer be maintained when its reference is not passed to us and neither are e returning head?

• Reply • Share ›

typing.. ➚ Nidhi • 2 years ago

by defining it as global variable, but this is also not allowed in this problem, so we can't change head pointer in any condition in this is problem..

• Reply • Share ›

Rakesh • 4 years ago

n = prev->next;
This line is missing before
prev->next = prev->next->next;

• Reply • Share ›

abhishek • 5 years ago

```
 package sam;

import java.util.ArrayList;
import java.util.List;

public class am {

        public static void main(String[] args) {
                //Generate linked list with 19->20->3->70->4->5
                List<Integer> n = new ArrayList<Integer>();
                n.add(19);
                n.add(20);
                n.add(3);
                n.add(70);
                n.add(4);
                n.add(5);
                LinkedList formed = generateLinkedList(n);
                deleteNode(formed.next.next.next);
```

see more

• Reply • Share ›

jogi • 5 years ago

this question was asked in Oravle interview.

1
• Reply • Share ›

ahmet alp balkan • 5 years ago

I think you should state that this linked list is singly-linked.

• Reply • Share ›

Chummi ➚ ahmet alp balkan • 5 years ago

interviewers generally do not mention it. If they say linked list they mean singly

• Reply • Share ›

GeeksforGeeks ➚ ahmet alp balkan • 5 years ago

@ahmet alp balkan: Thaks for the suggestion. We have added Singly to the problem statement.

• Reply • Share ›

trying_to_learn ➚ GeeksforGeeks • 5 years ago

Hi,

I understand the algorithm and tried to run it in Visual Studio and it runs well with the desired output but I do not understand this line in the deleteNode function.

// store address of next node
n = head->next;

Followed by free(n).

This means that n points to the next node which is 15 in this case and then we say free(n) which means free the memory to which n is pointing at i.e 15. The program correctly deletes the node 12 which is the head node but I did not understand how it deleted 12 instead of 15

Any help would be highly appreciated.

Thank you,

Best
A

```
/* Paste your code here (You may delete these lines if not writing
```

• Reply • Share ›

amitp49 ➚ trying_to_learn • 5 years ago

@trying_to_learn
Here the main logic is we can not change the head pointer. so we can not simply move head to next node. Rather what it does is it copy the data of next node to head. so now first and second both node have the same data as of second node.

So in fact we reach to the state where we have deleted 12 logically but created 15 twice. Thats y it delete the second node to reach to final answer state.

Hope it helps u...

```
/* Copy the data of next node to head */
        head->data = head->next->data;

        // store address of next node
        n = head->next;
```

```
        // Remove the link of next node
        head->next = head->next->next;

        // free memory
        free(n);
```

2 • Reply • Share ›

Manish ➚ amitp49 • 4 years ago

Need one help. If I copy the head node to some temp node and mode my head pointer to head->next, then my head is pointing to the second node in the linklist. Now I just have to set the temp->next = null to delete the previous head link. And If I am using C#, then since there is no ref to the prev node, memory will be reclaimed by GC whenever it is required.

Node temp = head->next;
head = head->next;
temp->next = null;

Can some one tell me what is wrong with the above lines?

```
/* Paste your code here (You may delete these lines if not w
```