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;
}
```

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
```

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
```