Delete a Node

locked

Problem | Submissions | Leaderboard | Discussions

This challenge is part of a tutorial track by MyCodeSchool and is accompanied by a video lesson.

Submissions: 133
Max Score: 25
Difficulty: Easy

Rate This Challenge:

More

You're given the pointer to the head node of a linked list and the position of a node to delete. Delete the node at the given position and return the head node. A position of 0 indicates head, a position of 1 indicates one node away from the head and so on. The list may become empty after you delete the node.

Input Format

You have to complete the `deleteNode(SinglyLinkedListNode* llist, int position)` method which takes two arguments - the head of the linked list and the position of the node to delete. You should NOT read any input from stdin/console. `position` will always be at least 0 and less than the number of the elements in the list.

The first line of input contains an integer n, denoting the number of elements in the linked list.
The next n lines contain an integer each in a new line, denoting the elements of the linked list in the order.
The last line contains an integer $position$ denoting the position of the node that has to be deleted form the linked list.

Constraints

- $1 \leq n \leq 1000$
- $1 \leq list_i \leq 1000$, where $list_i$ is the i^{th} element of the linked list.

Output Format

Delete the node at the given position and `return` the head of the updated linked list. Do NOT print anything to stdout/console.

The code in the editor will print the updated linked list in a single line separated by spaces.

Sample Input

```
8
20
6
2
19
7
4
15
9
3
```

Sample Output

```
20 6 2 7 4 15 9
```

Explanation

The given linked list is `20->6->2->19->7->4->15->9`. We have to delete the node at position 3, which is 19. After deleting that node, the updated linked list is: `20->6->2->7->4->15->9`

Current Buffer (saved locally, editable) C++

```
#include <cmath>

using namespace std;

class SinglyLinkedListNode {
    public:
        int data;
        SinglyLinkedListNode *next;

        SinglyLinkedListNode(int node_data) {
            this->data = node_data;
            this->next = nullptr;
        }
};

class SinglyLinkedList {
    public:
        SinglyLinkedListNode *head;
        SinglyLinkedListNode *tail;

        SinglyLinkedList() {
            this->head = nullptr;
            this->tail = nullptr;
        }

        void insert_node(int node_data) {
            SinglyLinkedListNode* node = new SinglyLinkedListNode(node_data);

            if (!this->head) {
                this->head = node;
            } else {
                this->tail->next = node;
            }

            this->tail = node;
        }
};

void print_singly_linked_list(SinglyLinkedListNode* node, string sep, ofstream& fout) {
    while (node) {
        fout << node->data;

        node = node->next;

        if (node) {
            fout << sep;
        }
    }
}

void free_singly_linked_list(SinglyLinkedListNode* node) {
    while (node) {
        SinglyLinkedListNode* temp = node;
        node = node->next;

        free(temp);
    }
}

// Complete the deleteNode function below.

/*
 * For your reference:
 *
 * SinglyLinkedListNode {
 *     int data;
 *     SinglyLinkedListNode* next;
 * };
 *
 */
SinglyLinkedListNode* deleteNode(SinglyLinkedListNode* head, int position) {
    if(position==0){
        head=head->next;
        return head;
    }
    SinglyLinkedListNode* prev=new SinglyLinkedListNode(1);
    SinglyLinkedListNode* cur=new SinglyLinkedListNode(1);
    cur=head;
    for(int i=0;i<position;i++){
        prev=cur;
        cur=cur->next;
    }
prev->next=cur->next;
    return head;
}

int main()
{
    ofstream fout(getenv("OUTPUT_PATH"));

    SinglyLinkedList* llist = new SinglyLinkedList();

    int llist_count;
    cin >> llist_count;
    cin.ignore(numeric_limits<streamsize>::max(), '\n');

    for (int i = 0; i < llist_count; i++) {
        int llist_item;
        cin >> llist_item;
        cin.ignore(numeric_limits<streamsize>::max(), '\n');

        llist->insert_node(llist_item);
    }

    int position;
    cin >> position;
    cin.ignore(numeric_limits<streamsize>::max(), '\n');

    SinglyLinkedListNode* llist1 = deleteNode(llist->head, position);

    print_singly_linked_list(llist1, " ", fout);
    fout << "\n";

    free_singly_linked_list(llist1);

    fout.close();

    return 0;
}
```

Line: 28 Col: 1

Upload Code as File | Test against custom input | Run Code | Submit Code