Delete last occurrence of an item from linked list

Given a liked list and a key to be deleted. Delete last occurrence of key from linked. The list may have duplicates.

Examples:

```
Input:   1->2->3->5->2->10, key = 2
Output:  1->2->3->5->10
```

We strongly recommend you to minimize your browser and try this yourself first

The idea is to traverse the linked list from beginning to end. While traversing, keep track of last occurrence key. After traversing the complete list, delete last occurrence by copying data of next node and deleting the next node.

```
// A C++ program to demonstrate deletion of last
// Node in singly linked list
#include<bits/stdc++.h>

// A linked list Node
struct Node
{
    int key;
    struct Node *next;
};

void deleteLast(Node* head, int key)
{
    // Initialize previous of Node to be deleted
    Node* x = NULL;

    // Start from head and find the Node to be
    // deleted
    Node* temp = head;
    while (temp)
    {
        // If we found the key, update xv
        if (temp->key == key)
            x = temp;

        temp = temp->next;
    }

    // key occurs atleast once
    if (x != NULL)
    {
        // Copy key of next Node to x
        x->key = x->next->key;

        // Store and unlink next
        temp = x->next;
        x->next = x->next->next;

        // Free memorey for next
        delete temp;
    }
}

/* Utility function to create a new node with
   given key */
Node *newNode(int key)
{
    Node *temp = new Node;
    temp->key = key;
    temp->next = NULL;
    return temp;
}
```

```
// This function prints contents of linked list
// starting from the given Node
void printList(struct Node *node)
{
    while (node != NULL)
    {
        printf(" %d ", node->key);
        node = node->next;
    }
}

/* Drier program to test above functions*/
int main()
{
    /* Start with the empty list */
    struct Node* head = newNode(1);
    head->next = newNode(2);
    head->next->next = newNode(3);
    head->next->next->next = newNode(5);
    head->next->next->next->next =
                                   newNode(2);
    head->next->next->next->next->next =
                                  newNode(10);

    puts("Created Linked List: ");
    printList(head);
    deleteLast(head, 2);
    puts("\nLinked List after Deletion of 1: ");
    printList(head);
    return 0;
}
```

Output:

```
Created Linked List:
 1  2  3  5  2  10
Linked List after Deletion of 1:
 1  2  3  5  10
```

The above solution doesn't work when the node to be deleted is the last node.

Following solution handles all cases.

```
// A C++ program to demonstrate deletion of last
// Node in singly linked list
#include<bits/stdc++.h>

// A linked list Node
struct Node
{
    int key;
    struct Node *next;
};

void deleteLast(Node** head, int key)
{
    // Initialize previous of Node to be deleted
    Node* x = NULL;

    // Start from head and find the previous node of the Node to be
    // deleted
    Node* temp = *head;
    while (temp)
    {
        // If we found the key, update x
        if (temp->next && temp->next->key == key)
            x = temp;

        temp = temp->next;
    }

    // key occurs atleast once except head node
    if (x)
    {
```

```
        // Store and unlink next
        temp = x->next;
        x->next = x->next->next;

        // Free memorey for next
        delete temp;
    }
    else
    {
        if (*head && (*head)->key == key )
        {
            Node * temp = *head;
            *head =  (*head)->next;
            delete temp;
        }
    }
}

/* Utility function to create a new node with
given key */
Node *newNode(int key)
{
    Node *temp = new Node;
    temp->key = key;
    temp->next = NULL;
    return temp;
}

// This function prints contents of linked list
// starting from the given Node
void printList(struct Node *node)
{
    while (node != NULL)
    {
        printf(" %d ", node->key);
        node = node->next;
    }
}

/* Drier program to test above functions*/
int main()
{
    /* Start with the empty list */
    struct Node* head = newNode(1);
    head->next = newNode(2);
    head->next->next = newNode(3);
    head->next->next->next = newNode(5);
    head->next->next->next->next =
        newNode(2);
    head->next->next->next->next->next =
        newNode(1);

    puts("Created Linked List: ");
    printList(head);
    deleteLast(&head, 2);
    puts("\nLinked List after Deletion of 2: ");
    printList(head);
    return 0;
}
```

Output:

```
Created Linked List:
 1  2  3  5  2  10
Linked List after Deletion of 1:
 1  2  3  5  10
```

Source: http://qa.geeksforgeeks.org/6583/write-function-deletes-occurrence-specific-value-linked-list