Delete N nodes after M nodes of a linked list

Given a linked list and two integers M and N. Traverse the linked list such that you retain M nodes then delete next N nodes, continue the same till end of the linked list.

Difficulty Level: Rookie

Examples:

```
Input:
M = 2, N = 2
Linked List: 1->2->3->4->5->6->7->8
Output:
Linked List: 1->2->5->6

Input:
M = 3, N = 2
Linked List: 1->2->3->4->5->6->7->8->9->10
Output:
Linked List: 1->2->3->6->7->8

Input:
M = 1, N = 1
Linked List: 1->2->3->4->5->6->7->8->9->10
Output:
Linked List: 1->3->5->7->9
```

The main part of the problem is to maintain proper links between nodes, make sure that all corner cases are handled. Following is C implementation of function skipMdeleteN() that skips M nodes and delete N nodes till end of list. It is assumed that M cannot be 0.

C

```
// C program to delete N nodes after M nodes of a linked list
#include <stdio.h>
#include <stdlib.h>

// A linked list node
struct node
{
    int data;
    struct node *next;
};

/* Function to insert a node at the beginning */
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;
}

/* Function to print linked list */
void printList(struct node *head)
{
    struct node *temp = head;
```

```c
    struct node *temp = head;
    while (temp != NULL)
    {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}

// Function to skip M nodes and then delete N nodes of the linked list.
void skipMdeleteN(struct node  *head, int M, int N)
{
    struct node *curr = head, *t;
    int count;

    // The main loop that traverses through the whole list
    while (curr)
    {
        // Skip M nodes
        for (count = 1; count<M && curr!= NULL; count++)
            curr = curr->next;

        // If we reached end of list, then return
        if (curr == NULL)
            return;

        // Start from next node and delete N nodes
        t = curr->next;
        for (count = 1; count<=N && t!= NULL; count++)
        {
            struct node *temp = t;
            t = t->next;
            free(temp);
        }
        curr->next = t; // Link the previous list with remaining nodes

        // Set current pointer for next iteration
        curr = t;
    }
}

// Driver program to test above functions
int main()
{
    /* Create following linked list
      1->2->3->4->5->6->7->8->9->10 */
    struct node* head = NULL;
    int M=2, N=3;
    push(&head, 10);
    push(&head, 9);
    push(&head, 8);
    push(&head, 7);
    push(&head, 6);
    push(&head, 5);
    push(&head, 4);
    push(&head, 3);
    push(&head, 2);
    push(&head, 1);

    printf("M = %d, N = %d \nGiven Linked list is :\n", M, N);
    printList(head);

    skipMdeleteN(head, M, N);

    printf("\nLinked list after deletion is :\n");
    printList(head);

    return 0;
}
```

Python

```
# Python program to delete Mnodes after N nodes

# Node class
class Node:

    # Constructor to initialize the node object
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:

    # Function to initialize head
    def __init__(self):
        self.head = None

    # Function to insert a new node at the beginning
    def push(self, new_data):
        new_node = Node(new_data)
        new_node.next = self.head
        self.head = new_node

    # Utility function to prit the linked LinkedList
    def printList(self):
        temp = self.head
        while(temp):
            print temp.data,
            temp = temp.next

    def skipMdeleteN(self, M, N):
        curr = self.head

        # The main loop that traverses through the
        # whole list
        while(curr):
            # Skip Mnodes
            for count in range(1, M):
                if curr is None:
                    return
                curr = curr.next

            if curr is None :
                return

            # Start from next node and delete N nodes
            t = curr.next
            for count in range(1, N+1):
                if t is None:
                    break
                t = t.next

            # Link the previous list with reamining nodes
            curr.next = t
            # Set Current pointer for next iteration
            curr = t

# Driver program to test above function

# Create following linked list
# 1->2->3->4->5->6->7->8->9->10
llist = LinkedList()
M = 2
N = 3
llist.push(10)
llist.push(9)
llist.push(8)
llist.push(7)
llist.push(6)
llist.push(5)
llist.push(4)
llist.push(3)
llist.push(2)
llist.push(1)

print "M = %d, N = %d\nGiven Linked List is:" %(M, N)
llist.printList()
print

llist.skipMdeleteN(M, N)

print "\nLinked list after deletion is"
llist.printList()

# This code is contributed by Nikhil Kumar Singh(nickzuck_007)
```

Output:

```
M = 2, N = 3
Given Linked list is :
1 2 3 4 5 6 7 8 9 10

Linked list after deletion is :
1 2 6 7
```

Time Complexity: O(n) where n is number of nodes in linked list.