Move last element to front of a given Linked List

Write a C function that moves last element to front in a given Singly Linked List. For example, if the given Linked List is 1->2->3->4->5, then the function should change the list to 5->1->2->3->4.

Algorithm:
Traverse the list till last node. Use two pointers: one to store the address of last node and other for address of second last node. After the end of loop do following operations.
i) Make second last as last (secLast->next = NULL).
ii) Set next of last as head (last->next = *head_ref).
iii) Make last as head (*head_ref = last)

C/C++

```
/* C Program to move last element to front in a given linked list */
#include<stdio.h>
#include<stdlib.h>

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

/* We are using a double pointer head_ref here because we change
   head of the linked list inside this function.*/
void moveToFront(struct node **head_ref)
{
    /* If linked list is empty, or it contains only one node,
      then nothing needs to be done, simply return */
    if (*head_ref == NULL || (*head_ref)->next == NULL)
        return;

    /* Initialize second last and last pointers */
    struct node *secLast = NULL;
    struct node *last = *head_ref;

    /*After this loop secLast contains address of second last
    node and last contains address of last node in Linked List */
    while (last->next != NULL)
    {
        secLast = last;
        last = last->next;
    }

    /* Set the next of second last as NULL */
    secLast->next = NULL;

    /* Set next of last as head node */
    last->next = *head_ref;

    /* Change the head pointer to point to last node now */
    *head_ref = last;
}

/* UTILITY FUNCTIONS */
/* Function to add a node at the begining of Linked List */
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 nodes in a given linked list */
void printList(struct node *node)
{
    while(node != NULL)
    {
        printf("%d ", node->data);
        node = node->next;
    }
}

/* Druver program to test above function */
int main()
{
    struct node *start = NULL;

    /* The constructed linked list is:
     1->2->3->4->5 */
    push(&start, 5);
    push(&start, 4);
    push(&start, 3);
    push(&start, 2);
    push(&start, 1);

    printf("\n Linked list before moving last to front\n");
    printList(start);

    moveToFront(&start);

    printf("\n Linked list after removing last to front\n");
    printList(start);

    return 0;
}
```

Java

```
/* Java Program to move last element to front in a given linked list */
class LinkedList
{
    Node head;  // head of list

    /* Linked list Node*/
    class Node
    {
        int data;
        Node next;
        Node(int d) {data = d; next = null; }
    }

    void moveToFront()
    {
        /* If linked list is empty or it contains only
           one node then simply return. */
           if(head == null || head.next == null)
              return;

        /* Initialize second last and last pointers */
        Node secLast = null;
        Node secLast = null;
        Node last = head;

        /* After this loop secLast contains address of
           second last  node and last contains address of
           last node in Linked List */
        while (last.next != null)
        {
           secLast = last;
           last = last.next;
        }

        /* Set the next of second last as null */
        secLast.next = null;

        /* Set the next of last as head */
        last.next = head;

        /* Change head to point to last node. */
        head = last;
    }


    /* Utility functions */

    /* Inserts a new Node at front of the list. */
    public void push(int new_data)
    {
        /* 1 & 2: Allocate the Node &
                  Put in the data*/
        Node new_node = new Node(new_data);

        /* 3. Make next of new Node as head */
        new_node.next = head;

        /* 4. Move the head to point to new Node */
        head = new_node;
    }

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

     /* Drier program to test above functions */
    public static void main(String args[])
    {
        LinkedList llist = new LinkedList();
        /* Constructed Linked List is 1->2->3->4->5->null */
        llist.push(5);
        llist.push(4);
        llist.push(3);
        llist.push(2);
        llist.push(1);

        System.out.println("Linked List before moving last to front ");
        llist.printList();

        llist.moveToFront();

        System.out.println("Linked List after moving last to front ");
        llist.printList();
    }
}
/* This code is contributed by Rajat Mishra */
```

Output:

```
 Linked list before moving last to front
1 2 3 4 5
 Linked list after removing last to front
5 1 2 3
```

Time Complexity: O(n) where n is the number of nodes in the given Linked List.