GeeksforGeeks

A computer science portal for geeks

Practice IDE Q&A GeeksQuiz

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;
}
```

Run on IDE

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 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 */
```

Run on IDE

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.

Please write comments if you find any bug in above code/algorithm, or find other ways to solve the same problem.

Querying with Transact-SQL

edX edx.org

Enroll now

Self-Paced

52 Comments Category: Linked Lists

Related Posts:

- Merge two sorted linked lists such that merged list is in reverse order
- Compare two strings represented as linked lists
- Rearrange a given linked list in-place.
- Sort a linked list that is sorted alternating ascending and descending orders?
- Select a Random Node from a Singly Linked List
- Merge Sort for Doubly Linked List
- Point to next higher value node in a linked list with an arbitrary pointer
- Swap nodes in a linked list without swapping data

(Login to Rate and Mark)

1.2 Average Difficulty : **1.2/5.0**
Based on **4** vote(s)

☐ Add to TODO List
☐ Mark as DONE

Like Share Be the first of your friends to like this.

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.

52 Comments GeeksforGeeks 1 Login

Recommend Share Sort by Newest

Join the discussion…

Aryan Singhania • a month ago

There is one more way to do this just keep copying the data from one node to the next until you reach the end, save the data of the end node in a temp variable and copy this temp to first node data.

Pseudo Code :

```
circularShiftByOneLinkedList(Node head){
Node current = head;
int temp;
while(current != null){
temp = current->next->data;
current->next->data = current->data;
current = current->next;
}
head->data = temp;
}
```

• Reply • Share ›

Jatin • 2 months ago

```
public void moveLastToTop() {
Node node = HEAD;
Node prevToLast = HEAD;
while( node.getNext() != null){
prevToLast = node;
node = node.getNext();
}
prevToLast.setNext(HEAD);
node.setNext(HEAD.getNext());
HEAD.setNext(null);
HEAD = node;
}
```

• Reply • Share ›

Sree Ranjini • 4 months ago

```
void moveToFront(Node*& head)
{
Node* curr = head;
if (curr == NULL || curr->next == NULL) return;
while (curr && curr->next)
{
if (curr->next->next == NULL) break;
curr = curr->next;
}
curr->next->next = head;
head=curr->next;
curr->next = NULL;
}
```

^ | ˅ • Reply • Share ›

Rams • 4 months ago

```
void movelast(node** head)

{

struct node *temp=*head;

while(temp->next->next)

temp=temp->next;

struct node *FirstNode=new node;

FirstNode->key=temp->next->key;

FirstNode->next=*head;

*head=FirstNode;

temp->next=NULL;

}
```

^ | ˅ • Reply • Share ›

Abhishek Kumar • 5 months ago

plzz tell what is the problem with this code. My system crashes after running the program

```
void movelast(node** head)
{
if(*head == NULL || (*head)->next == NULL)
return;
node *secondlast=*head;
node *last= secondlast->next;
while (secondlast->next!=NULL)
{
secondlast=secondlast->next;
last=last->next;
}

last->next=*head;
```

*head=last;
}

• Reply • Share ›

Eknoor Abhishek Kumar • 5 months ago
the while condition is not correct. use last->next!=NULL instead

• Reply • Share ›

rishabh goel • 6 months ago
1.traverse till last
2.temp = last_node->data
3.last_node->data=head->data
4.head->data = temp

• Reply • Share ›

Ankur Chauhan • 6 months ago
Is there someting wrong with this?

```
void moveToFront(struct node **head_ref)
{

if(*head_ref == NULL || (*head_ref)->next == NULL)
return;

while(current->next->next!=NULL)
{
current=current->next;
}

temp=current->next;
current->next=NULL; //Assuming temp is allocated node memory and other prerequsites
temp->next=head;
*head_ref=temp;
}
```

• Reply • Share ›

Rohit Saluja Ankur Chauhan • 5 months ago
Yes you have not defined current in above snippet and you are using.

• Reply • Share ›

Shailendra Anant • 6 months ago
/* can be done using recursion */

```
void movetofirst(struct node* start)
```

```
{
struct node* current = start;
if (current != NULL && current->next != NULL)
{
movetofirst(start->next);
swap(¤t->data, ¤t->next->data);
}
}
```

• Reply • Share ›

Rohit Saluja Shailendra Anant • 5 months ago

what is t here ?
You have not defined it anywhere

• Reply • Share ›

blank space • 6 months ago

https://ideone.com/ejqshp

• Reply • Share ›

abc123 • 6 months ago

```
void rotate_right(listnode *node)
{
if(node==NULL)
return;

rotate_right(node->next);
if(node->next)
swap(node->data,node->next->data);

}
```

• Reply • Share ›

Rohit Saluja abc123 • 5 months ago

I guess there is one issue with your code.
In function call swap you have passed the arguments by value but you should pass them by reference

• Reply • Share ›

Mridul Ranjan • 7 months ago

// can be done recursively and simply

```
struct node * moveToFront(struct node **root){

if((*root)->next->next == NULL){
```

```
struct node *temp = (*root)->next;

(*root)->next = NULL;

return temp;

}

struct node *last = moveToFront(&((*root)->next));

last->next = *root;

return last;

}
```

• Reply • Share ›

Hawkings Mridul Ranjan • 7 months ago

good one!!nice thinking..

• Reply • Share ›

Avinash Madhwani • 7 months ago

```
/*can be done using single pointer*/

struct node *last2front(struct node *head)

{

if(!head||!head->next)

return head;

struct node *temp=head;

while(temp->next->next)

{

temp=temp->next;

}

temp->next->next=head;

head=temp->next;

temp->next=NULL;

return head;
```

}

1 • Reply • Share ›

Rohit Saluja Avinash Madhwani • 5 months ago

Yes it can be done using one pointer but using the pros of using an extra pointer will be the increased readability of the code.

• Reply • Share ›

nishant gupta • 9 months ago

Using one extra pointer:
http://ideone.com/0csV3G

• Reply • Share ›

Ekluv • 10 months ago

My code

```
void movelast2first(node **head){

node* t=*head;

node* y;

while(t->next->next!=NULL){

t=t->next;

}

y=t->next;

t->next=NULL;

y->next=*head;

*head=y;

}
```

• Reply • Share ›

adasd • 10 months ago

adadas

• Reply • Share ›

devakar verma • 10 months ago

we can also swap data of first node and last node.

• Reply • Share ›

Anshuman Singh • a year ago

```
struct node * move(struct node * start)
{
struct node *p,*last;
p=start;
while(p->next->next!=NULL)
p=p->next;
last=p->next;
last->next=start;
start=last;
p->next=NULL;
return start;
}
```

• Reply • Share ›

Utkarsh Mishra • a year ago

```
node* move(node* head)
{
node *loop=head;
while(loop->link->link!=NULL)
loop=loop->link;
loop->link->link=head; //make circular
head=loop->link; //change head
loop->link=NULL; /break circle
return head;
}
```

• Reply • Share ›

Hinata Hyuga • a year ago

we can do it with single node.
1.traverse through the list till second last.
2.use temp pointer to store the last. and make second last->next = NULL.
3.temp->next = *head
4.*head = temp.

• Reply • Share ›

Guest • a year ago

```
int getcount(struct node *head)
{
struct node *curr=head;
int count=0;
while(curr!=NULL &&curr->next!=NULL)
```

```
{
count++;
curr=curr->next;
}
return count;
}

void movelast(struct node *head)
{
struct node *curr=head;
struct node *p=head;
struct node *q=head;
int n=getcount(head);
```

see more

• Reply • Share ›

Naval • a year ago

using ony one pointer no need to use 2 pointer last and second last we can done it using only second last pointer

here is the code

```
#include<stdio.h>
#include<stdlib.h>

struct node
{
int data;
struct node* next;
};

void moveFront(struct node **);

void push(struct node** head_ref, int new_data)
{
struct node* new_node =(struct node*) malloc(sizeof(struct node));
```

see more

• Reply • Share ›

RajaCEGian Naval • a year ago

Thank you

• Reply • Share ›

Vamsha vardhan reddy • a year ago

Algorithms:

1. traverse the whole list till last node.

2.swap the first node data with last node data.

code:

http://coliru.stacked-crooked....

• Reply • Share ›

happysshao • a year ago

I think the sample code is not correct.

1->2->3->4;

will output 4->1->2->3

• Reply • Share ›

neeraj kumar happysshao • a year ago

This is what needed!

• Reply • Share ›

Kim Jong-il • a year ago

Simple Code

```
struct node *MoveLastElementFirst(struct node *head)
{
struct node *next,*prev;

if(head==null || head->link == NULL)
return head;

prev=head;
next = head->link;
while(next->link!=NULL)
{
prev=next;
next = next->link;
}

prev->link=NULL;
next->link=head;
head = next;
return head;
```

}

• Reply • Share ›

Vivek • a year ago

head=reverse(head)
head->next=reverse(head->next)
i think this should do

1 • Reply • Share ›

Kim Jong-il Vivek • a year ago

I do not think it work first, if it works then its inefficient.

1 • Reply • Share ›

Jaiwardhan Swarnakar • a year ago

Go to the last node, link it with the main root node and return back recursively
here is the ideone link
http://ideone.com/qbfxsq

1 • Reply • Share ›

ALEX • 2 years ago

/*stop at second last node*/
while(secondLastNode->next->next)
secondLastNode=secondLastNode->next;
/*make linked list circuler*/
secondLastNode->next->next=head;
head=secondLastNode;
/*put a NULL */
secondLastNode->next=NULL;

• Reply • Share ›

wishall • 2 years ago

Bug:head node link should be made pointing 2 NULL,,,,
(*head_ref)->next=NULL; before *head_ref=last;

• Reply • Share ›

v3gA wishall • 2 years ago

No. We are not swapping head & the last node.

• Reply • Share ›

wishall v3gA • 2 years ago

yes,i was wrong,,,no bug

• Reply • Share ›

Akash Panda • 2 years ago

void MoveLastToFront(struct node **head)

{

struct node *current=*head;

if(current==NULL || current->next==NULL)

return;

while(current->next->next!=NULL)

{

current=current->next;

}

struct node *temp=current->next;

current->next=NULL;

temp->next=*head;

*head=temp;

}

• Reply • Share ›

Himanshu Dagar • 2 years ago

even we can do it with a single pointer by keep track of forward nodes frm current node

• Reply • Share ›

Guest • 2 years ago

void move_last_node_to_beg(struct node **head)
{
struct node **temp=&((*head)->link); //temp holds address of link part of 1st node which is pointed to by head node
if(*temp!=NULL) //this is just to handle the case that the 1ST node itself is not the last node
{ while(((*temp)->link)!=NULL) //find the address present in the link field of 1st node by *temp,then go to that address and check if that nodes link field is null then quit the loop
, temp=&((*temp)->link); //this is basically to make temp to hold the next node's link field's address
//finally temp will hold address of the last but 1 nodes's link field's address...bcoz while loop quits when the next nodes link field contains null

(*temp)->link=*head; //now change the address of the present in the last nodes link to make it point to the head node
*head=*temp; //head node now points to where earlier the last but 1 node's link field was pointing that is to the last node
*temp=NULL; //the last but 1 node's link field now contains null
}
}

• Reply • Share ›

adithya • 4 years ago

```
 /* Make last node first */
void reverse(node **head) {
        node *temp,*temp1;
        temp=*head;
        temp1=*head;
        temp=temp->link;
        while(temp1->link!=NULL) {
                temp=temp->link;
                temp1=temp1->link->link;
        }
        temp1->link=*head;
        *head=temp1;
        temp->link=NULL;
        return;
}
```

• Reply • Share ›

adithya • 4 years ago

```
 /* Function for making lastnode first*/
void reverse(node **head) {
        node *temp,*temp1;
        temp=*head;
        temp1=*head;
        temp=temp->link;
        while(temp1->link!=NULL) {
                temp=temp->link;
                temp1=temp1->link->link;
        }
        temp1->link=*head;
        *head=temp1;
        temp->link=NULL;
        return;
}
```

• Reply • Share ›

Venki • 4 years ago

Function to move last node to start of the list with only one crawl pointer. Comments explains the logic.

```
 void moveToFront(struct node **head_ref)
{
    /* Proceed only when list is valid (efficient code) */
    if( *head_ref && (*head_ref)->next )
    {
        struct node *ite = *head_ref;

        /* Move to second last node */
        while( ite && ite->next && ite->next->next )
        {
            ite = ite->next;
        }

        /* Make the list circular */
        ite->next->next = *head_ref;
        /* Set up new head */
        *head_ref = ite->next;
        /* Break the loop */
        ite->next= NULL;
    }
}
```

3 • Reply • Share ›

ashish jaiswal Venki • a year ago

venki...thats good...realy...

• Reply • Share ›

Murali S Iyengar Venki • 2 years ago

@Venki
The check "ite && ite->next" in the while loop is redundant as you have already checked for head and head->next in "if" at the beginning.

The while loop may be changed to

while (ite->next->next)
{

ite = ite->next;
}

1 • Reply • Share ›

renu Venki • 2 years ago

awesome!!!

• Reply • Share ›

Coder Venki • 3 years ago

Nice approach really good Venki :)

• Reply • Share ›

Soumya Sengupta Venki • 3 years ago

@venki-great iterative code......enjoyed it...

```
/* Paste your code here (You may delete these lines if not writing code) */
```

• Reply • Share ›

Load more comments