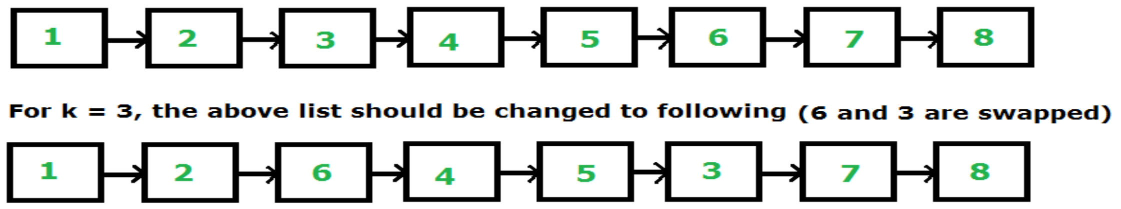
Swap Kth node from beginning with Kth node from end in a Linked List

Given a singly linked list, swap kth node from beginning with kth node from end. **Swapping of data is not allowed, only pointers should be changed.** This requirement may be logical in many situations where the linked list data part is huge (For example student details line Name, RollNo, Address, ..etc). The pointers are always fixed (4 bytes for most of the compilers).


1 2 3 4 5 6 7 8
For k = 3, the above list should be changed to following (6 and 3 are swapped)
1 2 6 4 5 3 7 8

The problem seems simple at first look, but it has many interesting cases.

Let X be the kth node from beginning and Y be the kth node from end. Following are the interesting cases that must be handled.
1) Y is next to X
2) X is next to Y
3) X and Y are same
4) X and Y don't exist (k is more than number of nodes in linked list)

We strongly recommend you to try it yourself first, then see the below solution. It will be a good exercise of pointers.

```
// A C++ program to swap Kth node from beginning with kth node from en
```

Google™ Custom Search

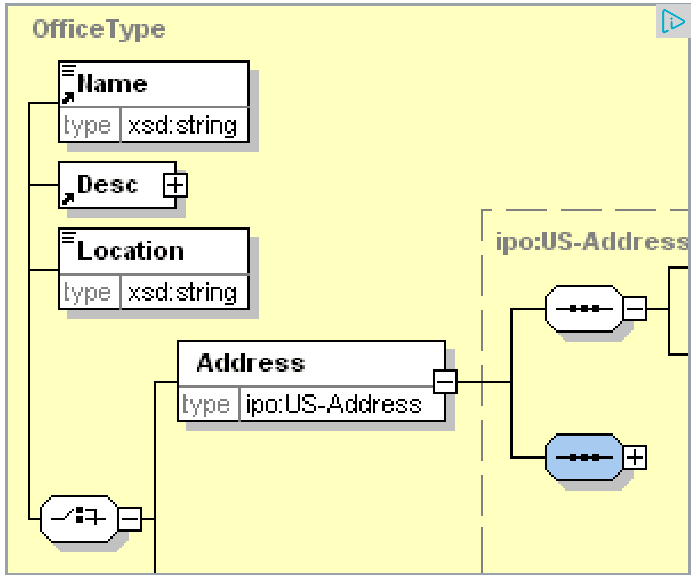
GeeksforGeeks

53,527 people like GeeksforGeeks.

Interview Experiences
Advanced Data Structures
Dynamic Programming
Greedy Algorithms
Backtracking
Pattern Searching
Divide & Conquer
Mathematical Algorithms
Recursion

```
#include <iostream>
#include <stdlib.h>
using namespace std;

// A Linked List node
struct node
{
    int data;
    struct node *next;
};

/* Utility function to insert a node at the beginning */
void push(struct node **head_ref, int new_data)
{
    struct node *new_node = (struct node *) malloc(sizeof(struct node)
    new_node->data = new_data;
    new_node->next = (*head_ref);
    (*head_ref) = new_node;
}

/* Utility function for displaying linked list */
void printList(struct node *node)
{
    while (node != NULL)
    {
        cout << node->data << " ";
        node = node->next;
    }
    cout << endl;
}

/* Utility function for calculating length of linked list */
int countNodes(struct node *s)
{
    int count = 0;
    while (s != NULL)
    {
        count++;
        s = s->next;
    }
    return count;
}

/* Function for swapping kth nodes from both ends of linked list */
void swapKth(struct node **head_ref, int k)
{
    // Count nodes in linked list
```

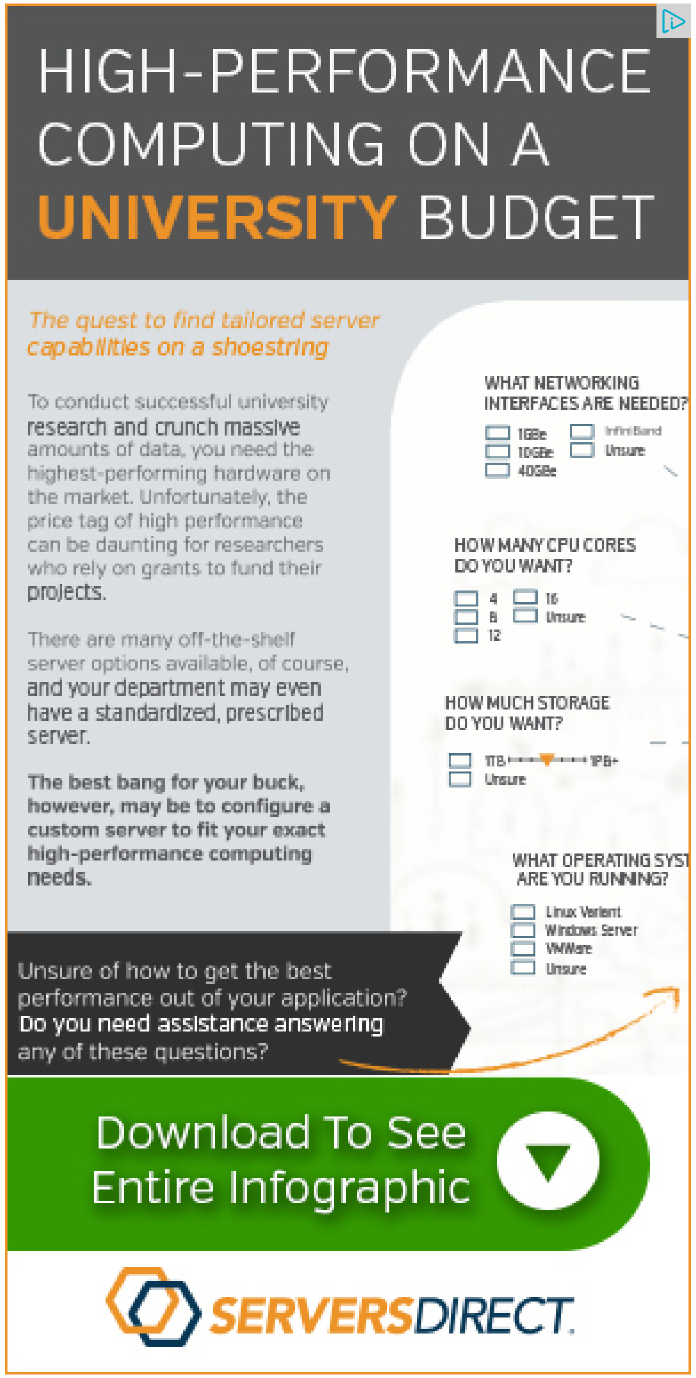
Geometric Algorithms



Popular Posts

- All permutations of a given string
- Memory Layout of C Programs
- Understanding “extern” keyword in C
- Median of two sorted arrays
- Tree traversal without recursion and without stack!
- Structure Member Alignment, Padding and Data Packing
- Intersection point of two Linked Lists
- Lowest Common Ancestor in a BST.
- Check if a binary tree is BST or not
- Sorted Linked List to Balanced BST

```
    int n = countNodes(*head_ref);

    // Check if k is valid
    if (n < k)  return;

    // If x (kth node from start) and y(kth node from end) are same
    if (2*k - 1 == n) return;

    // Find the kth node from beginning of linked list. We also find
    // previous of kth node because we need to update next pointer of
    // the previous.
    node *x = *head_ref;
    node *x_prev = NULL;
    for (int i = 1; i < k; i++)
    {
        x_prev = x;
        x = x->next;
    }

    // Similarly, find the kth node from end and its previous. kth nod
    // from end is (n-k+1)th node from beginning
    node *y = *head_ref;
    node *y_prev = NULL;
    for (int i = 1; i < n-k+1; i++)
    {
        y_prev = y;
        y = y->next;
    }

    // If x_prev exists, then new next of it will be y. Consider the c
    // when y->next is x, in this case, x_prev and y are same. So the
    // "x_prev->next = y" creates a self loop. This self loop will be
    // when we change y->next.
    if (x_prev)
        x_prev->next = y;

    // Same thing applies to y_prev
    if (y_prev)
        y_prev->next = x;

    // Swap next pointers of x and y. These statements also break self
    // loop if x->next is y or y->next is x
    node *temp = x->next;
    x->next = y->next;
    y->next = temp;

    // Change head pointers when k is 1 or n
```


HIGH-PERFORMANCE COMPUTING ON A UNIVERSITY BUDGET
The quest to find tailored server capabilities on a shoestring
To conduct successful university research and crunch massive amounts of data, you need the highest-performing hardware on the market. Unfortunately, the price tag of high performance can be daunting for researchers who rely on grants to fund their projects.
There are many off-the-shelf server options available, of course, and your department may even have a standardized, prescribed server.
The best bang for your buck, however, may be to configure a custom server to fit your exact high-performance computing needs.
WHAT NETWORKING INTERFACES ARE NEEDED?
HOW MANY CPU CORES DO YOU WANT?
HOW MUCH STORAGE DO YOU WANT?
WHAT OPERATING SYST ARE YOU RUNNING?
Unsure of how to get the best performance out of your application? Do you need assistance answering any of these questions?
Download To See Entire Infographic
SERVERSDIRECT

```
    if (k == 1)
        *head_ref = y;
    if (k == n)
        *head_ref = x;
}

// Driver program to test above functions
int main()
{
    // Let us create the following linked list for testing
    // 1->2->3->4->5->6->7->8
    struct node *head = NULL;
    for (int i = 8; i >= 1; i--)
       push(&head, i);

    cout << "Original Linked List: ";
    printList(head);

    for (int k = 1; k < 10; k++)
    {
        swapKth(&head, k);
        cout << "\nModified List for k = " << k << endl;
        printList(head);
    }

    return 0;
}
```

Output:

```
Original Linked List: 1 2 3 4 5 6 7 8

Modified List for k = 1
8 2 3 4 5 6 7 1

Modified List for k = 2
8 7 3 4 5 6 2 1

Modified List for k = 3
8 7 6 4 5 3 2 1
```

Recent Comments

Abhi You live US or India?

Google (Mountain View) interview · 46 minutes ago

Aman Hi, Why arent we checking for conditions...

Write a C program to Delete a Tree. · 1 hour ago

kzs please provide solution for the problem...

Backtracking | Set 2 (Rat in a Maze) · 1 hour ago

Sanjay Agarwal bool tree::Root_to_leaf_path_given_sum(tree...

Root to leaf path sum equal to a given number · 1 hour ago

GOPI GOPINATH @admin Highlight this sentence "We can easily...

Count trailing zeroes in factorial of a number · 1 hour ago

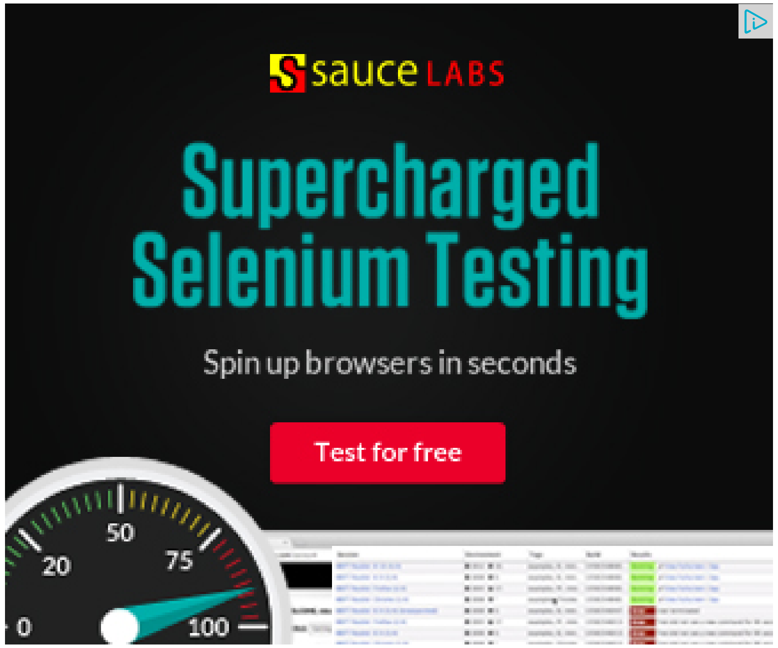
newCoder3006 If the array contains negative numbers also. We...

Find subarray with given sum · 2 hours ago

```
Modified List for k = 4
8 7 6 5 4 3 2 1

Modified List for k = 5
8 7 6 4 5 3 2 1

Modified List for k = 6
8 7 3 4 5 6 2 1

Modified List for k = 7
8 2 3 4 5 6 7 1

Modified List for k = 8
1 2 3 4 5 6 7 8

Modified List for k = 9
1 2 3 4 5 6 7 8
```

Please note that the above code runs three separate loops to count nodes, find x and x prev, and to find y and y_prev. These three things can be done in a single loop. The code uses three loops to keep things simple and readable.

Thanks to Chandra Prakash for initial solution. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.



Related Tpoics:

- Given a linked list, reverse alternate nodes and append at the end
- Pairwise swap elements of a given linked list by changing links
- Self Organizing List | Set 1 (Introduction)
- Merge a linked list into another linked list at alternate positions
- QuickSort on Singly Linked List
- Delete N nodes after M nodes of a linked list
- Design a stack with operations on middle element
- QuickSort on Doubly Linked List