# Arista Network

## Doubly linked list

Given a sorted doubly link list and two numbers C and K. You need to decrease the info of node with data K by C and insert the new node formed at its correct position such that the list remains sorted.  
Complete working code with test cases was required.

Sol:

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

struct Node{

int val;

Node \*prev, \*next;

Node(){

val=0;

prev=next=NULL;

}

Node(int v): val(v){

prev=next=NULL;

}

};

void print(Node \*head){

Node \*tmp=head;

while(tmp){

cout<<tmp->val<<" ";

tmp=tmp->next;

}

cout<<endl;

}

Node\* find(Node \*\*head, int val){

if(!\*head) return NULL;

Node \*h=\*head;

while(h->val < val)

h=h->next;

if(h->val == val) return h;

else return NULL;

}

void deleteN(Node \*\*head, Node \*node){

if(!\*head || !node) return;

if(\*head==node)

\*head=node->next;

if(node->next)

node->next->prev=node->prev;

if(node->prev)

node->prev->next=node->next;

}

void insertBefore(Node \*\*head, Node \*next, Node \*node){

if(!next) return;

node->prev=next->prev;

node->next=next;

next->prev=node;

if(node->prev)

node->prev->next=node;

else

\*head=node;

}

void insertAfter(Node \*prev, Node \*node){

if(!prev) return;

node->next=prev->next;

node->prev=prev;

prev->next=node;

if(node->next)

node->next->prev=node;

}

void solve(Node \*\*head, int C, int K){

if(!\*head) return;

Node \*node=find(head, C);

Node \*tmp;

if(!node) {

cout<<"Node with value "<<C<<" not found"<<endl;

return;

}

node->val-=K;

print(\*head);

if(node->prev && node->prev->val > node->val){

deleteN(head, node);

tmp=node->prev;

while(tmp && tmp->val>node->val)

tmp=tmp->prev;

if(!tmp)

insertBefore(head, \*head, node);

else

insertAfter(tmp, node);

return;

}

if(node->next && node->next->val < node->val){

deleteN(head, node);

tmp=node->next;

Node \*tail;

while(tmp && tmp->val < node->val){

tail=tmp;

tmp=tmp->next;

}

if(!tmp)

insertAfter(tail, node);

else

insertBefore(head, tmp, node);

return;

}

}

int main()

{

cout<<"Hello World"<<endl;

Node \*head=new Node(10);

head->next=new Node(20);

head->next->prev=head;

head->next->next=new Node(25);

head->next->next->prev=head->next;

head->next->next->next=new Node(35);

head->next->next->next->prev=head->next->next;

print(head);

solve(&head, 20, -6);

print(head);

return 0;

}

# VMWare

## [Longest Common Subsequence](https://practice.geeksforgeeks.org/problems/longest-common-subsequence/0)

## [Maximum Index](https://practice.geeksforgeeks.org/problems/maximum-index/0)

## [Array to BST](https://practice.geeksforgeeks.org/problems/array-to-bst/0)

## [Egg Dropping Puzzle](https://practice.geeksforgeeks.org/problems/egg-dropping-puzzle/0)

## [K’th smallest element](https://practice.geeksforgeeks.org/problems/kth-smallest-element/0)

int randomPartition(int arr[], int l, int r);

// This function returns k'th smallest element in arr[l..r] using

// QuickSort based method. ASSUMPTION: ELEMENTS IN ARR[] ARE DISTINCT

int kthSmallest(int arr[], int l, int r, int k)

{

    // If k is smaller than number of elements in array

    if (k > 0 && k <= r - l + 1)

    {

        // Partition the array around a random element and

        // get position of pivot element in sorted array

        int pos = randomPartition(arr, l, r);

        // If position is same as k

        if (pos-l == k-1)

            return arr[pos];

        if (pos-l > k-1)  // If position is more, recur for left subarray

            return kthSmallest(arr, l, pos-1, k);

        // Else recur for right subarray

        return kthSmallest(arr, pos+1, r, k-pos+l-1);

    }

    // If k is more than the number of elements in the array

    return INT\_MAX;

}

## [Check for BST](https://practice.geeksforgeeks.org/problems/check-for-bst/1)

## [Finding middle element in a linked list](https://practice.geeksforgeeks.org/problems/finding-middle-element-in-a-linked-list/1)

## [Reverse a linked list](https://practice.geeksforgeeks.org/problems/reverse-a-linked-list/1)

## [Detect Loop in linked list](https://practice.geeksforgeeks.org/problems/detect-loop-in-linked-list/1)

## [Run Length Encoding](https://practice.geeksforgeeks.org/problems/run-length-encoding/1)

## [Height of Binary Tree](https://practice.geeksforgeeks.org/problems/height-of-binary-tree/1)

## [Infix to Postfix](https://practice.geeksforgeeks.org/problems/infix-to-postfix/0)

**Infix to Postfix**

##### **Submissions:**[**14942**](https://practice.geeksforgeeks.org/problem_submissions.php?pid=1974)**Accuracy:**

48.37%

##### **Difficulty:**[**Medium**](https://practice.geeksforgeeks.org/Medium/0/0/)**Marks: 4**

Show Topic Tags

Company Tags [Amazon](https://practice.geeksforgeeks.org/company/Amazon/)  [Paytm](https://practice.geeksforgeeks.org/company/Paytm/)  [Samsung](https://practice.geeksforgeeks.org/company/Samsung/)  [VMWare](https://practice.geeksforgeeks.org/company/VMWare/)

* [**Problems**](https://practice.geeksforgeeks.org/problems/infix-to-postfix/0#problems)

Given an infix expression in the form of a string **str**. Conver this infix expression to postfix expression.

* **Infix expression:** The expression of the form a **op** b. When an operator is in-between every pair of operands.
* **Postfix expression:** The expression of the form a b **op**. When an operator is followed for every pair of operands.

**Input:**  
The first line of input contains an integer T denoting the number of test cases. The next T lines contains an infix expression.The expression contains all characters and ^,\*,/,+,-.

**Output:**  
For each testcase, in a new line, output the infix expression to postfix expression.

**Constraints:**  
1 <= T <= 100  
1 <= length of str <= 103

**Example:  
Input:**  
2  
a+b\*(c^d-e)^(f+g\*h)-i  
A\*(B+C)/D

**Output:**  
abcd^e-fgh\*+^\*+i-  
ABC+\*D/

#include <iostream>

#include<bits/stdc++.h>

using namespace std;

bool isOperand(char c){

if((c>='a'&&c<='z')||(c>='A'&&c<='Z'))

return true;

else return false;

}

int prec(char c){

switch(c){

case '+':

case '-':

return 1;

case '\*':

case '/':

return 2;

case '^':

return 3;

default:

return -1;

}

}

string postFix(string A){

stack<char>st;

st.push('N');

string res;

int n=A.size();

for(int i=0; i<n; i++){

if(isOperand(A[i]))

res.push\_back(A[i]);

else if(A[i]=='(')

st.push('(');

else if(A[i]==')'){

while(st.top()!='N' && st.top()!='('){

res.push\_back(st.top());

st.pop();

}

if(st.top()=='(')

st.pop();

}

else{

while(st.top()!='N' && prec(st.top())>=prec(A[i])){

res.push\_back(st.top());

st.pop();

}

st.push(A[i]);

}

//cout<<res<<endl;

}

while(st.top()!='N'){

res.push\_back(st.top());

st.pop();

}

return res;

}

## [Diameter of Binary Tree](https://practice.geeksforgeeks.org/problems/diameter-of-binary-tree/1)

## [Mirror Tree](https://practice.geeksforgeeks.org/problems/mirror-tree/1)

## [Boolean Matrix Problem](https://practice.geeksforgeeks.org/problems/boolean-matrix-problem/0)