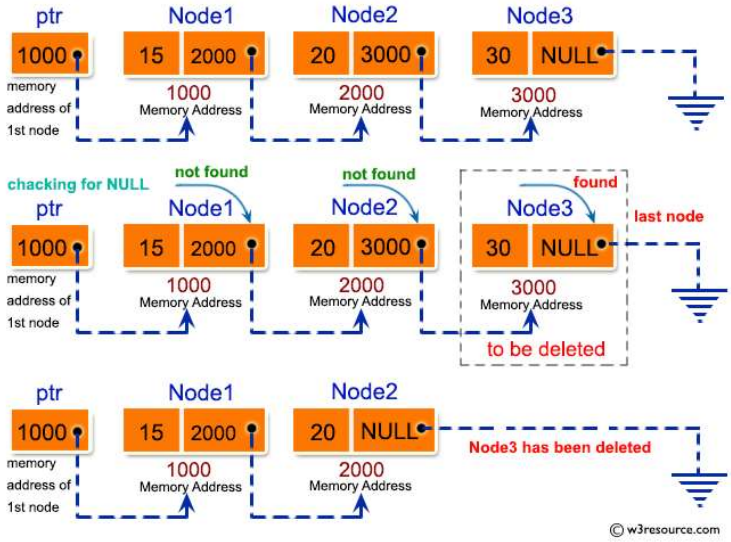
1- Write a program in C to delete the last node of Singly Linked List.




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

struct node
{
    int num;                    //Data of the node
    struct node *nextptr;       //Address of the node
}*stnode;

void createNodeList(int n);     //function to create the list
void LastNodeDeletion();       //function to delete the last nodes
void displayList();             //function to display the list

int main()
{
    int n,num,pos;
            printf("\n\n Linked List : Delete the last node of Singly Linked List :\n");
            printf("-----------------------------------------------------------------\n");

    printf(" Input the number of nodes : ");
    scanf("%d", &n);
    createNodeList(n);

    printf("\n Data entered in the list are : \n");
    displayList();
    LastNodeDeletion();
          printf("\n The new list after deletion the last node are  : \n");
    displayList();
    return 0;
}
```

```c
void createNodeList(int n)
{
    struct node *fnNode, *tmp;
    int num, i;

    stnode = (struct node *)malloc(sizeof(struct node));
    if(stnode == NULL) //check whether the stnode is NULL and if so no memory allocation
    {
        printf(" Memory cannot be allocated.");
    }
    else
    {
// reads data for the node through keyboard
        printf(" Input data for node 1 : ");
        scanf("%d", &num);

        stnode-> num = num;
        stnode-> nextptr = NULL; //Links the address field to NULL
        tmp = stnode;

//Creates n nodes and adds to linked list
        for(i=2; i<=n; i++)
        {
            fnNode = (struct node *)malloc(sizeof(struct node));
            if(fnNode == NULL) //check whether the fnnode is NULL and if so no memory allocation
            {
                printf(" Memory cannot be allocated.");
                break;
            }
            else
            {
                printf(" Input data for node %d : ", i);
                scanf(" %d", &num);
                fnNode->num = num;      // links the num field of fnNode with num
                fnNode->nextptr = NULL; // links the address field of fnNode with NULL
                tmp->nextptr = fnNode; // links previous node i.e. tmp to the fnNode
                tmp = tmp->nextptr;
            }
        }
    }
}
```

```c
// Deletes the last node of the linked list
void LastNodeDeletion()
{
    struct node *toDelLast, *preNode;
    if(stnode == NULL)
    {
        printf(" There is no element in the list.");
    }
    else
    {
        toDelLast = stnode;
        preNode = stnode;
        /* Traverse to the last node of the list*/
        while(toDelLast->nextptr != NULL)
        {
            preNode = toDelLast;
            toDelLast = toDelLast->nextptr;
        }
        if(toDelLast == stnode)
        {
            stnode = NULL;
        }
        else
        {

            /* Disconnects the link of second last node with last node */
            preNode->nextptr = NULL;
        }

        /* Delete the last node */
        free(toDelLast);
    }
}
// function to display the entire list
void displayList()
{
    struct node *tmp;
    if(stnode == NULL)
    {
        printf(" No data found in the empty list.");
    }
    else
    {
        tmp = stnode;
        while(tmp != NULL)
        {
            printf(" Data = %d\n", tmp->num);   // prints the data of current node
            tmp = tmp->nextptr;                 // advances the position of current node
        }
    }
}
```

```
Linked List : Delete the last node of Singly Linked List :
------------------------------------------------------------------
 Input the number of nodes : 3
 Input data for node 1 : 1
 Input data for node 2 : 2
 Input data for node 3 : 3

 Data entered in the list are :
 Data = 1
 Data = 2
 Data = 3

 The new list after deletion the last node are  :
 Data = 1
 Data = 2
```

2- Given a linked list and two keys in it, swap nodes for two given keys. Nodes should be swapped by changing links. Swapping data of nodes may be expensive in many situations when data contains many fields.

It may be assumed that all keys in linked list are distinct.

```
Input:  10->15->12->13->20->14,  x = 12, y = 20
Output: 10->15->20->13->12->14

Input:  10->15->12->13->20->14,  x = 10, y = 20
Output: 20->15->12->13->10->14

Input:  10->15->12->13->20->14,  x = 12, y = 13
Output: 10->15->13->12->20->14
```

```c
/* This program swaps the nodes of linked list rather
   than swapping the field from the nodes.*/

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

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

/* Function to swap nodes x and y in linked list by
   changing links */
void swapNodes(struct Node **head_ref, int x, int y)
{
   // Nothing to do if x and y are same
   if (x == y) return;

   // Search for x (keep track of prevX and CurrX
   struct Node *prevX = NULL, *currX = *head_ref;
   while (currX && currX->data != x)
   {
       prevX = currX;
       currX = currX->next;
   }

   // Search for y (keep track of prevY and CurrY
   struct Node *prevY = NULL, *currY = *head_ref;
   while (currY && currY->data != y)
   {
       prevY = currY;
       currY = currY->next;
   }

   // If either x or y is not present, nothing to do
   if (currX == NULL || currY == NULL)
       return;

   // If x is not head of linked list
   if (prevX != NULL)
       prevX->next = currY;
   else // Else make y as new head
       *head_ref = currY;

   // If y is not head of linked list
   if (prevY != NULL)
       prevY->next = currX;
   else  // Else make x as new head
       *head_ref = currX;

   // Swap next pointers
   struct Node *temp = currY->next;
   currY->next = currX->next;
   currX->next  = temp;
}
```

```c
/* Function to add a node at the begining of 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->6->7 */
    push(&start, 7);
    push(&start, 6);
    push(&start, 5);
    push(&start, 4);
    push(&start, 3);
    push(&start, 2);
    push(&start, 1);

    printf("\n Linked list before calling swapNodes() ");
    printList(start);

    swapNodes(&start, 4, 3);

    printf("\n Linked list after calling swapNodes() ");
    printList(start);

    return 0;
}
```