Write a function to delete a Linked List

Algorithm For C/C++: Iterate through the linked list and delete all the nodes one by one. Main point here is not to access next of the current pointer if current pointer is deleted.

In **Java**, automatic garbage collection happens, so deleting a linked list is easy. We just need to change head to null.

Implementation:

C/C++

```c
// C program to delete a linked list
#include<stdio.h>
#include<stdlib.h>
#include<assert.h>

/* Link list node */
struct node
{
    int data;
    struct node* next;
};

/* Function to delete the entire linked list */
void deleteList(struct node** head_ref)
{
   /* deref head_ref to get the real head */
   struct node* current = *head_ref;
   struct node* next;

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

   /* deref head_ref to affect the real head back
      in the caller. */
   *head_ref = NULL;
}

/* Given a reference (pointer to pointer) to the head
  of a list and an int, push a new node on the front
  of the 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;
}

/* Drier program to test count function*/
int main()
{
    /* Start with the empty list */
    struct node* head = NULL;

    /* Use push() to construct below list
     1->12->1->4->1  */
    push(&head, 1);
    push(&head, 4);
    push(&head, 1);
    push(&head, 12);
    push(&head, 1);

    printf("\n Deleting linked list");
    deleteList(&head);

    printf("\n Linked list deleted");
}
```

Java

```
// Java program to delete a linked list
class LinkedList
{
 Node head; // head of the list

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

    /* Function deletes the entire linked list */
 void deleteList()
 {
     head = null;
 }

 /* 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;
    }

    public static void main(String [] args)
    {
     LinkedList llist = new LinkedList();
     /* Use push() to construct below list
           1->12->1->4->1  */

        llist.push(1);
        llist.push(4);
        llist.push(1);
        llist.push(12);
        llist.push(1);

        System.out.println("Deleting the list");
        llist.deleteList();

        System.out.println("Linked list deleted");
    }
}
// This code is contributed by Rajat Mishra
```

Output:

```
 Deleting linked list
 Linked list deleted
```

Time Complexity: O(n)
Auxiliary Space: O(1)