

GeeksforGeeks

A computer science portal for geeks



**Suggest a
Topic**

Login

**Write an
Article**



Linked List |
Set 2
(Inserting a
node)

Reverse a
linked list

Stack Data
Structure
(Introduction
and
Program)

Detect loop
in a linked
list

Find the
middle of a
given linked
list in C and
Java

Detect and
Remove
Loop in a
Linked List

Linked List |
Set 3
(Deleting a
node)

Function to
check if a
singly linked
list is
palindrome

Merge Sort
for Linked
Lists

Reverse a
Linked List in
groups of
given size |
Set 1

Program for
n'th node
from the end
of a Linked
List

Doubly
Linked List |
Set 1
(Introduction
and
Insertion)

Delete a
Linked List
node at a
given
position

Merge two
sorted linked
lists

Write a
function to

get the
intersection
point of two
Linked Lists.

Linked List
vs Array

Remove
duplicates
from an
unsorted
linked list

Find Length
of a Linked
List (Iterative
and
Recursive)

Remove
duplicates
from a
sorted linked
list

Find length
of loop in
linked list

LinkedList in
Java

Circular
Linked List |
Set 1
(Introduction
and
Applications)

Swap nodes

in a linked
list without
swapping
data

Flattening a
Linked List

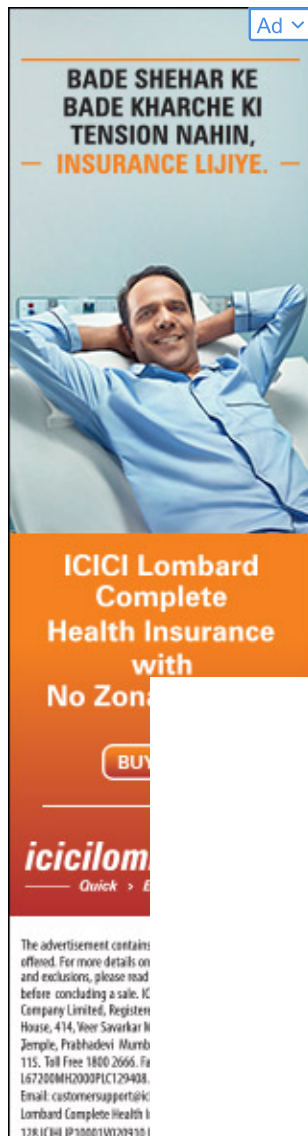
Add two
numbers
represented
by linked
lists | Set 2

Delete N
nodes after
M nodes of a
linked list

Top 20
Linked List
Interview
Question

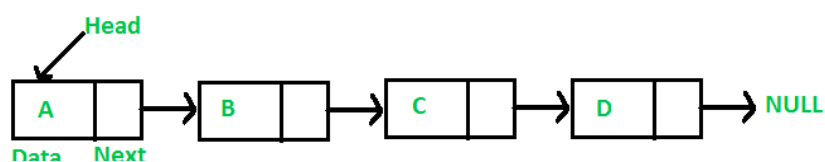
Add two
numbers
represented
by linked
lists | Set 1

Write a
function to
delete a
Linked List



Linked List | Set 1 (Introduction)

Like arrays, Linked List is a linear data structure. Unlike arrays, linked list elements are not stored at contiguous location; the elements are linked using pointers.



Why Linked List?

Arrays can be used to store linear data of similar types, but arrays have following limitations.

- 1) The size of the arrays is fixed: So we must know the upper limit on the number of elements in advance. Also, generally, the allocated memory is equal to the upper limit irrespective of the usage.
- 2) Inserting a new element in an array of elements is expensive, because room has to be created for the new elements and to create room existing elements have to be shifted.

Get world's best paying jobs - Learn AI & Blockchain

Enrol today for affordable courses in Emerging Technology

For example, in a system if we maintain a sorted list of IDs in an array `id[]`.

`id[] = [1000, 1010, 1050, 2000, 2040]`.

And if we want to insert a new ID 1005, then to maintain the sorted order, we have to move all the elements after 1000 (excluding 1000).

Deletion is also expensive with arrays unless some special techniques are used. For example, to delete 1010 in `id[]`, everything after 1010 has to be moved.

Advantages over arrays

- 1) Dynamic size
- 2) Ease of insertion/deletion

Get world's best paying jobs - Learn AI & Blockchain

Enrol today for affordable courses in Emerging Technology

Drawbacks:

- 1) Random access is not allowed. We have to access elements sequentially starting from the first node. So we cannot do binary search with linked lists efficiently with its default implementation. Read about it [here](#).
- 2) Extra memory space for a pointer is required with each element of the list.
- 3) Not cache friendly. Since array elements are contiguous locations, there is locality of reference which is not there in case of linked lists.

Representation:

A linked list is represented by a pointer to the first node of the linked list. The first node is called head. If the linked list is empty, then value of head is NULL.

Each node in a list consists of at least two parts:

- 1) data
- 2) Pointer (Or Reference) to the next node

In C, we can represent a node using structures. Below is an example of a linked list node with an integer data.

In Java, LinkedList can be represented as a class and a Node as a separate class. The LinkedList class contains a reference of Node class type.

C

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





Java

```
class LinkedList
{
    Node head; // head of list

    /* Linked list Node*/
    class Node
    {
        int data;
        Node next;

        // Constructor to create a new node
        // Next is by default initialized
        // as null
        Node(int d) {data = d;}
    }
}
```

Python



```
# Node class
class Node:

    # Function to initialize the node object
    def __init__(self, data):
        self.data = data # Assign data
        self.next = None # Initialize
                        # next as null

# Linked List class
class LinkedList:

    # Function to initialize the Linked
    # List object
    def __init__(self):
        self.head = None
```

First Simple Linked List in C Let us create a simple linked list with 3 nodes.

C


```

// A simple C program to introduce
// a linked list
#include<stdio.h>
#include<stdlib.h>

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

// Program to create a simple linked
// list with 3 nodes
int main()
{
    struct Node* head = NULL;
    struct Node* second = NULL;
    struct Node* third = NULL;

    // allocate 3 nodes in the heap
    head = (struct Node*)malloc(sizeof(struct Node));
    second = (struct Node*)malloc(sizeof(struct Node));
    third = (struct Node*)malloc(sizeof(struct Node));

    /* Three blocks have been allocated dynamically.
       We have pointers to these three blocks as first,
       second and third

           head           second           third
           |              |              |
       +---+---+---+   +---+---+---+   +---+---+---+
       | # | # |      | | # | # |      | | # | # |
       +---+---+---+   +---+---+---+   +---+---+---+

    # represents any random value.
    Data is random because we haven't assigned
    anything yet */





    head->data = 1; //assign data in first node
    head->next = second; // Link first node with
                        // the second node

    /* data has been assigned to data part of first
       block (block pointed by head). And next
       pointer of first block points to second.
       So they both are linked.

           head           second           third
           |              |              |
       +---+---+---+   +---+---+---+   +---+---+---+
       | 1 | 0----->| # | # |      | | # | # |
       +---+---+---+   +---+---+---+   +---+---+---+
    */

```

Java

```
// A simple Java program to introduce a linked list
class LinkedList
{
    Node head; // head of list

    /* Linked list Node. This inner class is made static so the
    main() can access it */
    static class Node {
        int data;
        Node next;
        Node(int d) { data = d; next=null; } // Constructor
    }

    /* method to create a simple linked list with 3 nodes*/
    public static void main(String[] args)
    {
        /* Start with the empty list. */
        LinkedList llist = new LinkedList();

        llist.head = new Node(1);
        Node second = new Node(2);
        Node third = new Node(3);

        /* Three nodes have been allocated dynamically.
        We have references to these three blocks as first,
        second and third

        llist.head          second          third
          |                  |              |
          |                  |              |
        +---+-----+      +---+-----+      +---+-----+
        | 1 | null |      | 2 | null |      | 3 | null |
        +---+-----+      +---+-----+      +---+-----+ */

        llist.head.next = second; // Link first node with the se

        /* Now next of first Node refers to second. So they
        both are linked.





        llist.head          second          third
          |                  |              |
          |                  |              |
        +---+-----+      +---+-----+      +---+-----+
        | 1 | 0----->| 2 | null |      | 3 | null |
        +---+-----+      +---+-----+      +---+-----+ */

        second.next = third; // Link second node with the third

        /* Now next of second Node refers to third. So all the
        nodes are linked.

        llist.head          second          third
          |                  |              |
          |                  |              |
        +---+-----+      +---+-----+      +---+-----+
        | 1 | 0----->| 2 | 0----->| 3 | null |
        +---+-----+      +---+-----+      +---+-----+ */
    }
}
```

Python

 # A simple Python program to introduce a linked list
 # Node class
 **class** Node:


```

    # Function to initialise the node object
    def __init__(self, data):
        self.data = data # Assign data
        self.next = None # Initialize next as null

# Linked List class contains a Node object
class LinkedList:

    # Function to initialize head
    def __init__(self):
        self.head = None

# Code execution starts here
if __name__ == '__main__':

    # Start with the empty list
    llist = LinkedList()

    llist.head = Node(1)
    second = Node(2)
    third = Node(3)

    ...

    Three nodes have been created.
    We have references to these three blocks as first,
    second and third

    llist.head          second          third
        |                |                |
        +---+-----+    +---+-----+    +---+-----+
        | 1 | None |    | 2 | None |    | 3 | None |
        +---+-----+    +---+-----+    +---+-----+
        ...

    llist.head.next = second; # Link first node with second
    ...

    Now next of first Node refers to second. So they
    both are linked.

    llist.head          second          third
        |                |                |
        +---+-----+    +---+-----+    +---+-----+
        | 1 | o----->| 2 | null |    | 3 | null |
        +---+-----+    +---+-----+    +---+-----+





```

Linked List Traversal

In the previous program, we have created a simple linked list with three nodes. Let us traverse the created list and print the data of each node. For traversal, let us write a general purpose function `printList()` that prints any given list.

We strongly recommend that you click here and practice it, before moving on to the solution.

C



```
// A simple C program for traversal of a linked list
#include<stdio.h>
#include<stdlib.h>

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

// This function prints contents of linked list starting from
// the given node
void printList(struct Node *n)
{
    while (n != NULL)
    {
        printf(" %d ", n->data);
        n = n->next;
    }
}

int main()
{
    struct Node* head = NULL;
    struct Node* second = NULL;
    struct Node* third = NULL;

    // allocate 3 nodes in the heap
    head = (struct Node*)malloc(sizeof(struct Node));
    second = (struct Node*)malloc(sizeof(struct Node));
    third = (struct Node*)malloc(sizeof(struct Node));

    head->data = 1; //assign data in first node
    head->next = second; // Link first node with second





    second->data = 2; //assign data to second node
    second->next = third;

    third->data = 3; //assign data to third node
    third->next = NULL;

    printList(head);

    return 0;
}
```

Java



```
// A simple Java program for traversal of a linked list
class LinkedList
{
    Node head; // head of list

    /* Linked list Node. This inner class is made static so the
    main() can access it */
    static class Node {
        int data;
        Node next;
        Node(int d) { data = d; next=null; } // Constructor
    }

    /* This function prints contents of linked list starting from
    head */
    public void printList()
    {
        Node n = head;
        while (n != null)
        {
            System.out.print(n.data+" ");
            n = n.next;
        }
    }





    /* method to create a simple linked list with 3 nodes */
    public static void main(String[] args)
    {
        /* Start with the empty list. */
        LinkedList llist = new LinkedList();

        llist.head = new Node(1);
        Node second = new Node(2);
        Node third = new Node(3);

        llist.head.next = second; // Link first node with the second
        second.next = third; // Link second node with the third

        llist.printList();
    }
}
```

Python

```
# A simple Python program for traversal of a linked list

# Node class
class Node:

    # Function to initialise the node object
    def __init__(self, data):
        self.data = data # Assign data
        self.next = None # Initialize next as null


# Linked List class contains a Node object
class LinkedList:

    # Function to initialize head
    def __init__(self):
        self.head = None

    # This function prints contents of linked list
    # starting from head
    def printList(self):
        temp = self.head
        while (temp):
            print temp.data,
            temp = temp.next


# Code execution starts here
if __name__ == '__main__':

    # Start with the empty list
    llist = LinkedList()

    llist.head = Node(1)
    second = Node(2)
    third = Node(3)

    llist.head.next = second; # Link first node with second
    second.next = third; # Link second node with the third node

    llist.printList()
```

Output:

1 2 3

Linked List | Set 1 (Introduction) | GeeksforGeeks




Important Links :

- [Practice MCQ Questions on Linked List](#)
- [Linked List Data Structure Page](#)
- [Coding Practice Questions on Linked List.](#)

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

StackPath DDoS Protection - Protect Your Edge

Complete DDoS protection designed to keep your website up and

 [ckpath.com/ddos-protection](https://stackpath.com/ddos-protection)

Recommended Posts:

[Find the middle of a given linked list in C and Java](#)

[Program for n'th node from the end of a Linked List](#)

[Write a function to get Nth node in a Linked List](#)

[Given only a pointer/reference to a node to be deleted in a singly linked list, how do you delete it?](#)

[Detect loop in a linked list](#)

[Write a function to delete a Linked List](#)

[Write a function that counts the number of times a given int occurs in a Linked List](#)

Reverse a linked list

Given only a pointer to a node to be deleted in a singly linked list, how do you delete it?

Write a function to get the intersection point of two Linked Lists.

Function to check if a singly linked list is palindrome

The Great Tree-List Recursion Problem.

Clone a linked list with next and random pointer | Set 1

Memory efficient doubly linked list

Given a linked list which is sorted, how will you insert in sorted way

Improved By : [ashwani khemani](#)

StackPath DDoS Protection - Protect Your Edge

Complete DDoS protection designed to keep your website up and

stackpath.com/ddos-protection

Article Tags : [Linked List](#)

Practice Tags : [Linked List](#)



16

1.5

☐ To-do ☐ Done

Based on **310** vote(s)

Feedback

Add Notes

Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

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

Share this post!

32 Comments GeeksforGeeks

 Login ▾ Recommend 8 Tweet Share

Sort by Newest ▾



Join the discussion...

LOG IN WITH

OR SIGN UP WITH DISQUS Name **ViralKahaani** • 18 days ago

I'm having problem with solving K reverse linked list problem from here -

<https://www.interviewbit.co...>

Can someone check and tell me?

^ | ▾ • Reply • Share >

**Namjith Aravind** • a month agoWhy asterisk is after "Node" in the following code. I am not familiar with c. Please help
struct Node* head =NULL;

5 ^ | ▾ • Reply • Share >

**amanta** → Namjith Aravind • 21 days ago

That's also my question

^ | ▾ • Reply • Share >

**Saurabh Tyagi** • 3 months ago

Any specific reason to make Node an inner class?

^ | ▾ • Reply • Share >

**Snehasish Ghosh** • 3 months ago

Simple Java

```
public class SimpleLinkedList {  
  
    static void printNode(LNode root) {  
        while(root != null) {  
            System.out.print(root.data + " ");  
            root = root.next;  
        }  
    }  
  
    public static void main(String[] args) {  
  
        LNode node_1 = new LNode(13);  
        LNode node_2 = new LNode(28);
```



710-B, Advant Navis Business Park,
Sector-142, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org

COMPANY

About Us
Careers
Privacy Policy
Contact Us

LEARN

Algorithms
Data Structures
Languages
CS Subjects
Video Tutorials

PRACTICE

Company-wise
Topic-wise
Contests
Subjective Questions

CONTRIBUTE

Write an Article
Write Interview
Experience
Internships
Videos

@geeksforgeeks, Some rights reserved