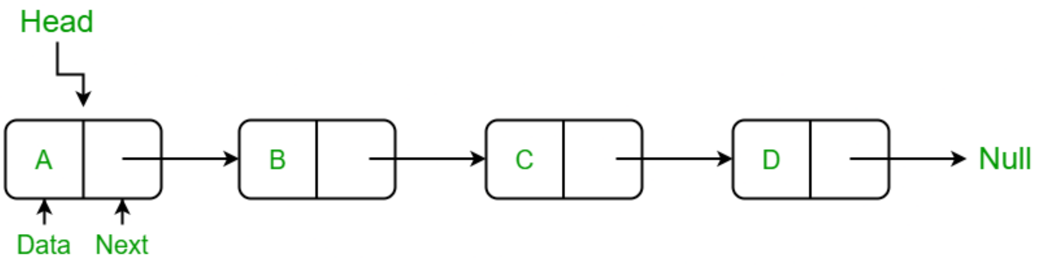
Nth Node from end of Linked List

Given a Linked List and a number **N**, write a function that returns the value at the Nth node from the end of the Linked List.



Linked-List

Examples:

> ***Input:*** *1 -> 2 -> 3 -> 4, N = 3*
> ***Output:*** *2*
>
> ***Input:*** *35 -> 15 -> 4 -> 20, N = 4*
> ***Output:*** *35*

Naive Approach: Follow the given steps to solve the problem using this approach:

- Calculate the length of the Linked List. Let the length be len.
- Print the (len – n + 1)th node from the beginning of the Linked List.

Below is the implementation of the above approach:

C++ Java

```
// Java program to find N'th node from
// end of linked list
class LinkedList {
    Node head; // head of the list
```

```
        int data;
            data = d;
            next = null;
        }
    }

    /* Function to get the Nth node from the last of a
    linked list */
    void printNthFromLast(int N)
    {
        int len = 0;
        Node temp = head;

        // 1) count the number of nodes in Linked List
        while (temp != null) {
            temp = temp.next;
            len++;
        }

        // check if value of N is not more than length of
        // the linked list
        if (len < N)
            return;

        temp = head;

        // 2) get the (len-N+1)th node from the beginning
        for (int i = 1; i < len - N + 1; i++)
            temp = temp.next;

        System.out.println(temp.data);
    }

    /* Inserts a new Node at front of the list. */
    public void push(int new_data)
    {
        /* 1 & 2: Allocate the Node &
                Put in the data*/
```

Dash
All
Articles
Videos
Problems
Quiz
Contest
Menu
Track Progress
69 of 132 Complete. (53%)

```
        /* 3. Make next of new Node as head */

        head = new_node;
    }

    // Driver's code
    public static void main(String[] args)
    {
        LinkedList llist = new LinkedList();
        llist.push(20);
        llist.push(4);
        llist.push(15);
        llist.push(35);

        // Function call
        llist.printNthFromLast(4);
    }
}
```

Dash
All
Articles
Videos
Problems
Quiz
Contest

Output

```
35
```

Time complexity: O(M) where M is the size of the linked list
Auxiliary Space: O(1)

Nth node from the end of a Linked List using two pointers:

> *As Nth node from the end equals to (Length – N + 1)th node from the start, so the idea is to Maintain two pointers starting from the head of the Linked-List and move one pointer to the Nth node from the start and then move both the pointers together until the pointer at the Nth position reaches the last node. Now the pointer which was moved later points at the Nth node from the end of the Linked-List*

Menu

Track Progress
69 of 132 Complete. (53%)

Dash

All

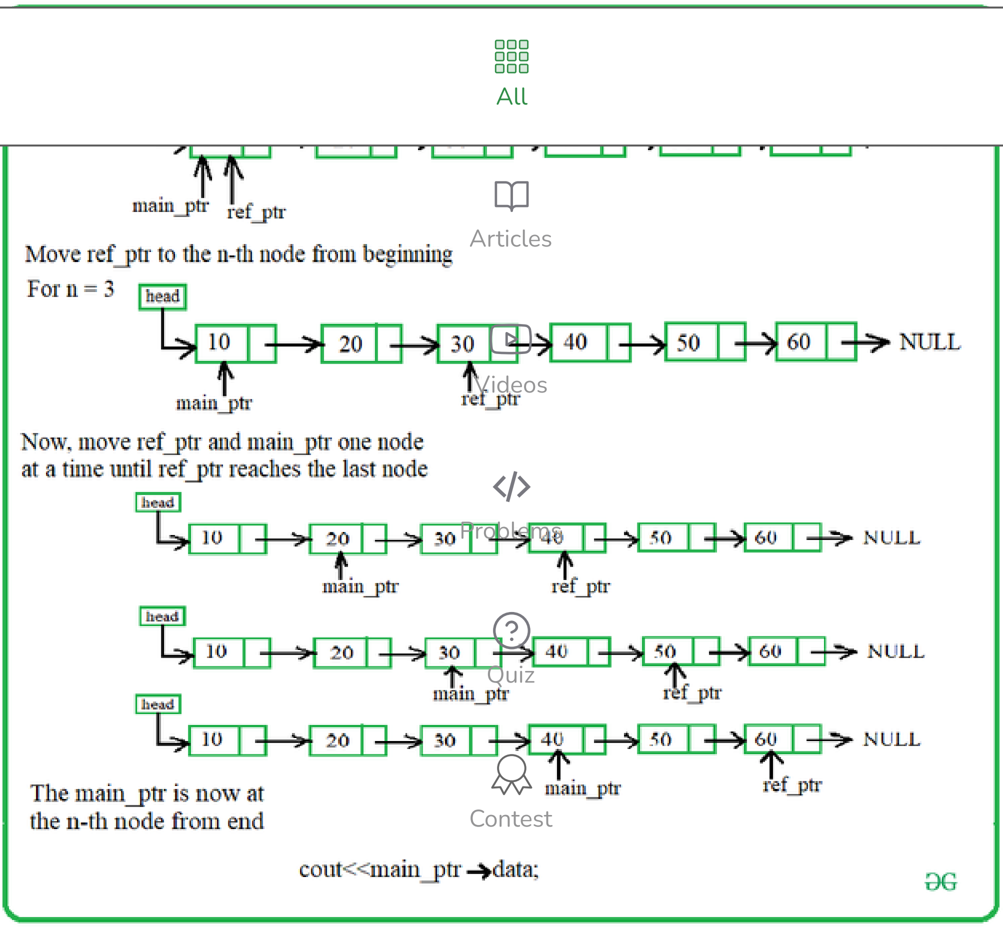
Articles

Videos

Problems

Quiz

Contest


main_ptr
ref_ptr
Move ref_ptr to the n-th node from beginning
For n = 3
head
10
20
30
40
50
60
NULL
main_ptr
ref_ptr
Now, move ref_ptr and main_ptr one node
at a time until ref_ptr reaches the last node
The main_ptr is now at
the n-th node from end
cout<<main_ptr→data;

Follow the given steps to solve the problem:

- Maintain two pointers main_ptr and ref_ptr
- Move ref_ptr to the Nth node from the start
- Now move both main_ptr and ref_ptr, until the ref_ptr reaches the last node
- Now print the data of the main_ptr, as it is at the Nth node from the end

Below is the implementation of the above approach:

C++ Java

```
 // Java program to find N'th
// node from end
class LinkedList {
    Node head; // head of the list

    /* Linked List node */
    class Node {
        int data;
```

Menu

Track Progress

69 of 132 Complete. (53%)

Dash
All
Articles
Videos
Problems
Quiz
Contest
Menu

```
    {
}

/* Function to get the
Nth node from end of list */
void printNthFromLast(int N)
{
    Node main_ptr = head;
    Node ref_ptr = head;

    int count = 0;
    if (head != null) {
        while (count < N) {
            if (ref_ptr == null) {
                System.out.println(
                    N + " is greater than the no "
                    + " of nodes in the list");
                return;
            }
            ref_ptr = ref_ptr.next;
            count++;
        }

        if (ref_ptr == null) {

            if (head != null)
                System.out.println("Node no. " + N
                                + " from last is "
                                + head.data);
        }
        else {

            while (ref_ptr != null) {
                main_ptr = main_ptr.next;
                ref_ptr = ref_ptr.next;
            }
            System.out.println("Node no. " + N
                            + " from last is "
```

Track Progress
69 of 132 Complete. (53%)

Dash
All
Articles
Videos
Problems
Quiz
Contest
90% Money-Back!
Courses
Tutorials
Jobs
Practice
Contests
P

```
        }

    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;
    }

    // Driver's code
    public static void main(String[] args)
    {
        LinkedList llist = new LinkedList();
        llist.push(20);
        llist.push(4);
        llist.push(15);
        llist.push(35);

        // Function call
        llist.printNthFromLast(4);
    }
}
```

Output

```
35->15->4->20->NULL
Node no. 4 from end is: 35
```

Menu

Time Complexity: O(M) where M is the length of the linked list.
Auxiliary Space: O(1)

Track Progress
69 of **132** Complete. (53%)

Dash

Mark as Read

All

Report An Issue

If you are facing any issue on this page. Please let us know.

Articles

Videos

Problems

Quiz

Contest

Menu

Track Progress

69 of **132** Complete. (53%)