Find n’th node from the end of a Linked List

Given a Linked List and a number n, write a function that returns the value at the n’th node from end of the Linked List.

Method 1 (Use length of linked list)

1) Calculate the length of Linked List. Let the length be len.

2) Print the (len – n + 1)th node from the begining of the Linked List.

C

```
// Simple C program to find n'th node from end
#include<stdio.h>
#include<stdlib.h>

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

/* Function to get the nth node from the last of a linked list*/
void printNthFromLast(struct node* head, int n)
{
    int len = 0, i;
    struct 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 (n-len+1)th node from the begining
    for (i = 1; i < len-n+1; i++)
       temp = temp->next;
```

```
    printf ("%d", temp->data);

    return;
}

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 above function*/
int main()
{
  /* Start with the empty list */
  struct node* head = NULL;

  // create linked 35->15->4->20
  push(&head, 20);
  push(&head, 4);
  push(&head, 15);
  push(&head, 35);

  printNthFromLast(head, 5);
  return 0;
}
```

Run on IDE

Java

```
// Simple Java program to find n'th node from end of 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 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 (n-len+1)th node from the begining
        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*/
        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;
    }

    /*Drier program to test above methods */
    public static void main(String [] args)
    {
        LinkedList llist = new LinkedList();
        llist.push(20);
        llist.push(4);
        llist.push(15);
        llist.push(35);

        llist.printNthFromLast(4);
    }
}// This code is contributed by Rajat Mishra
```

Run on IDE

Output:

```
35
```

Following is a recursive C code for the same method. Thanks to Anuj Bansal for providing following code.

```
void printNthFromLast(struct node* head, int n)
```

```
{
    static int i = 0;
    if (head == NULL)
       return;
    printNthFromLast(head->next, n);
    if (++i == n)
       printf("%d", head->data);
}
```

Run on IDE

Time Complexity: O(n) where n is the length of linked list.

Method 2 (Use two pointers)

Maintain two pointers – reference pointer and main pointer. Initialize both reference and main pointers to head. First move reference pointer to n nodes from head. Now move both pointers one by one until reference pointer reaches end. Now main pointer will point to nth node from the end. Return main pointer.

Implementation:

C

```
// C program to find n'th node from end using slow and
// fast pointers
#include<stdio.h>
#include<stdlib.h>

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

/* Function to get the nth node from the last of a linked list*/
void printNthFromLast(struct node *head, int n)
{
  struct node *main_ptr = head;
  struct node *ref_ptr = head;

  int count = 0;
  if(head != NULL)
  {
     while( count < n )
     {
        if(ref_ptr == NULL)
        {
           printf("%d is greater than the no. of "
                    "nodes in list", n);
           return;
        }
        ref_ptr = ref_ptr->next;
        count++;
     } /* End of while*/

     while(ref_ptr != NULL)
```

```
     {
        main_ptr = main_ptr->next;
        ref_ptr  = ref_ptr->next;
     }
     printf("Node no. %d from last is %d ",
             n, main_ptr->data);
  }
}

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 above function*/
int main()
{
  /* Start with the empty list */
  struct node* head = NULL;
  push(&head, 20);
  push(&head, 4);
  push(&head, 15);

  printNthFromLast(head, 3);
}
```

Run on IDE

Java

```
// Java program to find n'th node from end using slow and
// fast pointers
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 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++;
            }
            while (ref_ptr != null)
            {
                main_ptr = main_ptr.next;
                ref_ptr = ref_ptr.next;
            }
            System.out.println("Node no. "+n+" from last is "+
                               main_ptr.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*/
        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;
    }

    /*Drier program to test above methods */
    public static void main(String [] args)
    {
        LinkedList llist = new LinkedList();
        llist.push(20);
        llist.push(4);
        llist.push(15);
        llist.push(35);

        llist.printNthFromLast(4);
    }
} // This code is contributed by Rajat Mishra
```

Run on IDE

Output:

```
Node no. 4 from last is 35
```

Time Complexity: O(n) where n is the length of linked list.

Please write comments if you find the above codes/algorithms incorrect, or find other ways to solve the same problem.

122 Comments Category: Linked Lists Tags: Linked Lists

Related Posts:

- Merge two sorted linked lists such that merged list is in reverse order
- Compare two strings represented as linked lists
- Rearrange a given linked list in-place.
- Sort a linked list that is sorted alternating ascending and descending orders?
- Select a Random Node from a Singly Linked List
- Merge Sort for Doubly Linked List
- Point to next higher value node in a linked list with an arbitrary pointer
- Swap nodes in a linked list without swapping data

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