Write a function to get Nth node in a Linked List

Write a GetNth() function that takes a linked list and an integer index and returns the data value stored in the node at that index position.

Example:

```
Input:  1->10->30->14,  index = 2
Output: 30
The node at index 2 is 30
```

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

Algorithm:

```
1. Initialize count = 0
2. Loop through the link list
     a. if count is equal to the passed index then return current
         node
     b. Increment count
     c. change current to point to next of the current.
```

Implementation:

C

```c
// C program to find n'th node in linked list
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>

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

/* 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;
}

/* Takes head pointer of the linked list and index
    as arguments and return data at index*/
int GetNth(struct node* head, int index)
{
    struct node* current = head;
    int count = 0; /* the index of the node we're currently
                  looking at */
    while (current != NULL)
    {
       if (count == index)
          return(current->data);
       count++;
       current = current->next;
    }

    /* if we get to this line, the caller was asking
       for a non-existent element so we assert fail */
    assert(0);
}

/* Drier program to test above 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);

    /* Check the count function */
    printf("Element at index 3 is %d", GetNth(head, 3));
    getchar();
}
```

Java

```
// Java program to find n'th node in linked list

class Node
{
    int data;
    Node next;
    Node(int d)
    {
        data = d;
        next = null;
    }
}

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

    /* Takes index as argument and return data at index*/
    public int GetNth(int index)
    {
        Node current = head;
        int count = 0; /* index of Node we are
                          currently looking at */
        while (current != null)
        {
            if (count == index)
                return current.data;
            count++;
            current = current.next;
        }

        /* if we get to this line, the caller was asking
        for a non-existent element so we assert fail */
        assert(false);
        return 0;
    }

    /* Given a reference to the head of a list and an int,
       inserts a new Node on the front of the list. */
    public void push(int new_data)
    {

        /* 1. alloc the Node and put data*/
        Node new_Node = new Node(new_data);

        /* 2. Make next of new Node as head */
        new_Node.next = head;

        /* 3. Move the head to point to new Node */
        head = new_Node;
    }

    /* Drier program to test above functions*/
    public static void main(String[] args)
    {
        /* Start with empty list */
        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);

        /* Check the count function */
        System.out.println("Element at index 3 is "+llist.GetNth(3));
    }
}
```

Python

```
# A complete working Python program to find n'th node
# in 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 is in LinkedList class. It inserts
    # a new node at the beginning of Linked List.
    def push(self, new_data):

        # 1 & 2: Allocate the Node &
        #     Put in the data
        new_node = Node(new_data)

        # 3. Make next of new Node as head
        new_node.next = self.head

        # 4. Move the head to point to new Node
        self.head = new_node

    # Returns data at given index in linked list
    def getNth(self, index):
        current = self.head # Initialise temp
        count = 0 # Index of current node

        # Loop while end of linked list is not reached
        while (current):
            if (count == index):
                return current.data
            count += 1
            current = current.next

        # if we get to this line, the caller was asking
        # for a non-existent element so we assert fail
        assert(false)
        return 0;

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

    llist = 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);

    n = 3
    print ("Element at index 3 is :", llist.getNth(n))
```

Output:

```
Element at index 3 is 4
```

Time Complexity: O(n)