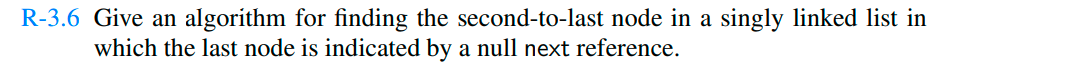
Try answering three of the following:



class Node

{

    int data;

    Node next;

    Node(int d)

    {

        this.data = d;

        this.next = null;

    }

}

class LinkedList

{

    Node start;

    LinkedList()

    {

        start = null;

    }

    // Function to push node at head

    public void push(int data)

    {

        if(this.start == null)

        {

        Node temp = new Node(data);

        this.start = temp;

        }

        else

        {

            Node temp = new Node(data);

            temp.next = this.start;

            this.start = temp;

        }

    }

    // method to find the second last

    // node of the linked list

    public int findSecondLastNode(Node ptr)

    {

        Node temp = ptr;

        // If the list is empty or contains less

        // than 2 nodes

        if(temp == null || temp.next == null)

            return -1;

            // This loop stops at second last node

        while(temp.next.next != null)

        {

            temp = temp.next;

        }

        return temp.data;

    }

    // Driver code

    public static void main(String[] args)

    {

        LinkedList ll = new LinkedList();

        /\* Use push() function to construct

        the below list 8 -> 23 -> 11 -> 29 -> 12 \*/

        ll.push(12);

        ll.push(29);

        ll.push(11);

        ll.push(23);

        ll.push(8);

        System.out.println(ll.findSecondLastNode(ll.start));

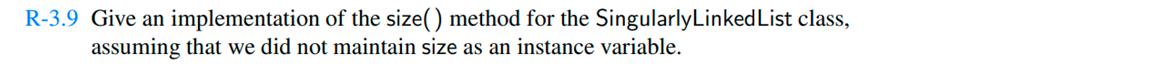
    }

}

**Output:**

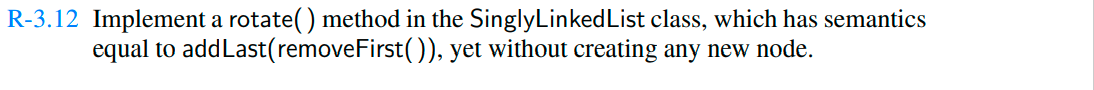
29

**Time complexity :** O(n)



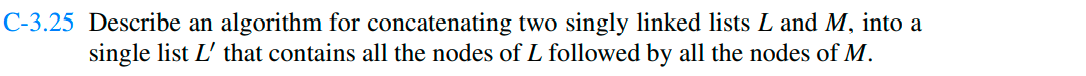
public int SizeA()  
{  
 Node<E> temp = head ;  
 int Size = 0 ;  
 if(head==null)  
 return Size ;  
 else  
 {  
 Size++ ;  
 while (temp.next!=null)  
 {  
 Size++;  
 temp = temp.next ;  
 }  
 }  
 return Size ;  
}

public class testsizeA {  
   
 public static void main(String[] args) {  
 SinglyLinkedList<Integer> list =new SinglyLinkedList<>();  
 list.addLast(11);  
 list.addLast(12);  
 list.addLast(13);  
 list.addLast(14);  
 System.*out*.println("The Size is : "+list.SizeA());  
 }  
   
}



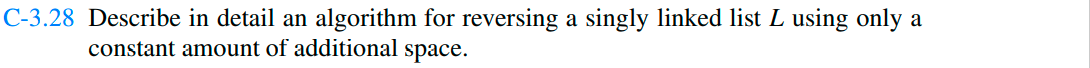
public void Rotate() {  
 if (head != null) {  
 Node current = head;  
 while (current.getNext() != null) {  
 tail = current;  
 current = current.getNext();  
 }  
 if (tail != null) {  
 tail.setNext(null);  
 current.setNext(head);  
 head = current;  
 }  
 }  
}

public static void main(String[] args) {  
 SinglyLinkedList<Integer> list =new SinglyLinkedList<>();  
 list.addLast(11);  
 list.addLast(12);  
 list.addLast(13);  
 list.addLast(14);  
 System.*out*.println(list.print());  
 list.Rotate();  
 System.*out*.println("After Rotation : \n");  
 System.*out*.println(list.print());  
}



public Node<E> concatenation(E head1 , E head2)  
{  
 Node<E> temp = null ;  
 if (head1==null)  
 return (Node<E>) head2;  
 if (head2==null)  
 return (Node<E>) head1;  
 temp= head.getNext();  
 while (temp.getNext()!=null)  
 temp = temp.next;  
 temp.next= tail.getNext();  
 return (Node<E>) head1;  
}

public class testconcatenation {  
 public static void main(String[] args) {  
 SinglyLinkedList<Integer> list = new SinglyLinkedList<>();  
 list.addLast(1);  
 list.addLast(2);  
 list.addLast(3);  
 list.addLast(4);  
 SinglyLinkedList<Integer> list2 = new SinglyLinkedList<>();  
 list.addLast(11);  
 list.addLast(12);  
 list.addLast(13);  
 list.addLast(14);  
 list.concatenation(list.first(),list2.last());  
 System.*out*.println(list.print());  
 }  
}



public Node<E> Reverse()  
{  
 if (head==null)  
 {  
 return head;  
 }  
 Node<E> current = head ;  
 Node<E> previous = null ;  
 Node<E> next = current.next ;  
  
  
 while (current!=null)  
 {  
 next = current.next ;  
 current.next=previous ;  
 previous = current ;  
 current = next ;  
 }  
  
 head = previous ;  
 return previous ;  
}

public class Task5 {  
 public static void main(String[] args) {  
 SinglyLinkedList<Integer> list =new SinglyLinkedList<>();  
 list.addLast(11);  
 list.addLast(12);  
 list.addLast(13);  
 list.addLast(14);  
 System.*out*.println(list.print());  
 System.*out*.println("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");  
 list.Reverse();  
 System.*out*.println(list.print());  
 }  
   
}

آية أبو الرجال

CS