**Lab5**

R-3.12 Implement a rotate( ) method in the SinglyLinkedList class, which has semantics equal to addLast(removeFirst( )), yet without creating any new node.

قم بتطبيق طريقة

rotate () method in SinglyLinkedList class

والتي تحتوي على دلالات

Equal to addLast (removeFirst ())

ولكن دون إنشاء أي عقدة جديدة .

public class Rotate {  
 static class LinkedList {  
 Node head;   
 /\* Linked list Node\*/  
 class Node {  
 int data;  
 Node next;  
 Node(int d)  
 {  
 data = d;  
 next = null;  
 }  
 }  
 // This function rotates a linked list counter-clockwise  
 // and updates the head. The function assumes that k is  
 // smaller than size of linked list. It doesn't modify  
 // the list if k is greater than or equal to size  
 void rotate(int k)  
 {  
 if (k == 0)  
 return;  
  
 // Let us understand the below code for example k = 4  
 // and list = 10->20->30->40->50->60.  
 Node current = head;  
  
 // current will either point to kth or NULL after this  
 // loop. current will point to node 40 in the above example  
 int count = 1;  
 while (count < k && current != null) {  
 current = current.next;  
 count++;  
 }  
  
 // If current is NULL, k is greater than or equal to count  
 // of nodes in linked list. Don't change the list in this case  
 if (current == null)  
 return;  
  
 // current points to kth node. Store it in a variable.  
 // kthNode points to node 40 in the above example  
 Node kthNode = current;  
  
 // current will point to last node after this loop  
 // current will point to node 60 in the above example  
 while (current.next != null)  
 current = current.next;  
  
 // Change next of last node to previous head  
 // Next of 60 is now changed to node 10  
  
 current.next = head;  
  
 // Change head to (k+1)th node  
 // head is now changed to node 50  
 head = kthNode.next;  
  
 // change next of kth node to null  
 kthNode.next = null;  
 }  
  
 /\* 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(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;  
 }  
  
 void printList()  
 {  
 Node temp = head;  
 while (temp != null) {  
 System.*out*.print(temp.data + " ");  
 temp = temp.next;  
 }  
 System.*out*.println();  
 }  
  
   
 public static void main(String args[])  
 {  
 LinkedList llist = new LinkedList();  
  
 // create a list 10->20->30->40->50->60  
 for (int i = 60; i >= 10; i -= 10)  
 llist.push(i);  
  
 System.*out*.println("Given list");  
 llist.printList();  
  
 llist.rotate(4);  
  
 System.*out*.println("Rotated Linked List");  
 llist.printList();  
 }  
 }   
}