package com.greatlearning.singlyLinkedList;

class Node {

public int nodeData;

public Node nextNode;

}

public class SinglyLinkedList {

private Node head;

public void displayLinkedList() {

Node currentNode = head;

while (currentNode != null) {

System.out.print(currentNode.nodeData + " -> ");

currentNode = currentNode.nextNode;

}

System.out.println("NULL");

}

public void insertAtFront(int data) {

Node newNode = new Node();

newNode.nodeData = data;

newNode.nextNode = head;

head = newNode;

}

public void insertAtLast(int data) {

Node currentNode = head;

while (currentNode.nextNode != null) {

currentNode = currentNode.nextNode;

}

Node newNode = new Node();

newNode.nodeData = data;

currentNode.nextNode = newNode;

}

public void deleteAtFront() {

head = head.nextNode;

}

public void deleteAtLast() {

Node currentNode = head;

if (head.nextNode == null) {

head = null;

} else {

while (currentNode.nextNode.nextNode != null) {

currentNode = currentNode.nextNode;

}

currentNode.nextNode = null;

}

}

public void insertAtPos(int data, int pos) {

Node currentNode = head;

for (int i = 0; i < pos - 1; i++) {

currentNode = currentNode.nextNode;

}

Node tempNode = currentNode;

currentNode = currentNode.nextNode;

Node newNode = new Node();

newNode.nodeData = data;

tempNode.nextNode = newNode;

newNode.nextNode = currentNode;

}

void deleteAtPos(int position) {

if (head == null)

return;

Node temp = head;

if (position == 1) {

head = temp.nextNode;

return;

}

// Find the previous node of the node that has to be deleted

for (int i = 1; temp != null && i < position - 1; i++)

temp = temp.nextNode;

if (temp == null || temp.nextNode == null) {

System.out.println("position entered is more than the no of elements present in the list");

return;

}

Node next = temp.nextNode.nextNode;

temp.nextNode = next;

}

public static void main(String args[]) {

SinglyLinkedList linkedList = new SinglyLinkedList();

linkedList.insertAtFront(10);

linkedList.insertAtFront(9);

linkedList.insertAtFront(1);

//

// linkedList.insertAtLast(6);

//

// Node node = new Node();

// node.nodeData = 9;

//

//

// Node node2 = new Node();

// node2.nodeData = 80;

//// linkedList.deleteAfter(node2);

//

// linkedList.deleteAtFront();

// linkedList.insertAtFront(17);

//

// linkedList.insertAtLast(44);

//

// linkedList.deleteAtFront();

//

// linkedList.insertAtLast(58);

//

// linkedList.deleteAtLast();

//

// linkedList.displayLinkedList();

// linkedList.insertAtPos(99, 2);

//

// linkedList.displayLinkedList();

//

// linkedList.insertAtPos(999, 1);

//

// linkedList.displayLinkedList();

// linkedList.insertAtPos(777, 6);

linkedList.displayLinkedList();

linkedList.displayLinkedList();

linkedList.displayLinkedList();

linkedList.deleteAtPos(3);

linkedList.displayLinkedList();

linkedList.deleteAtPos(2);

linkedList.displayLinkedList();

linkedList.deleteAtPos(1);

linkedList.displayLinkedList();

linkedList.deleteAtPos(1);

linkedList.displayLinkedList();

}

}