**Испит септември 2023**

Дадена е двострано поврзана листа и број k. Да се рутира листата во десно/лево k пати.

ЛЕВО:

**Input:**  
5  
1 2 3 4 5  
2

**Output:**  
3<->4<->5<->1<->2

|  |  |
| --- | --- |
| Input | Output |
| 5  0 2 4 6 8  6 | 2<->4<->6<->8<->0 |
| 3  0 1 2  4 | 1<->2<->0 |
| 6  10 20 30 40 50 60  3 | 40<->50<->60<->10<->20<->30 |
| 4  1 3 5 7  4 | 1<->3<->5<->7 |
| 10  10 20 30 40 50 60 70 80 90 100  6 | 70<->80<->90<->100<->10<->20<->30<->40<->50<->60 |
| 6  1 2 3 1 2 3  3 | 1<->2<->3<->1<->2<->3 |
| 2  3 7  2 | 3<->7 |
| 8  3 6 9 12 15 18 21 24  4 | 15<->18<->21<->24<->3<->6<->9<->12 |
| 4  0 0 0 0  3 | 0<->0<->0<->0 |

**Иницијален код:**

**import** java.io.BufferedReader;

**import** java.io.IOException;

**import** java.io.InputStreamReader;

**class** DLLNode<E> {

**protected** E element;

**protected** DLLNode<E> pred, succ;

**public** DLLNode(E elem, DLLNode<E> pred, DLLNode<E> succ) {

**this**.element = elem;

**this**.pred = pred;

**this**.succ = succ;

}

@Override

**public** String toString() {

**return** element.toString();

}

}

**class** DLL<E> {

**private** DLLNode<E> first, last;

**public** DLL() {

// Construct an empty SLL

**this**.first = **null**;

**this**.last = **null**;

}

**public** **void** deleteList() {

first = **null**;

last = **null**;

}

**public** **int** length() {

**int** ret;

**if** (first != **null**) {

DLLNode<E> tmp = first;

ret = 1;

**while** (tmp.succ != **null**) {

tmp = tmp.succ;

ret++;

}

**return** ret;

} **else**

**return** 0;

}

**public** DLLNode<E> find(E o) {

**if** (first != **null**) {

DLLNode<E> tmp = first;

**while** (tmp.element != o && tmp.succ != **null**)

tmp = tmp.succ;

**if** (tmp.element == o) {

**return** tmp;

} **else** {

System.***out***.println("Elementot ne postoi vo listata");

}

} **else** {

System.***out***.println("Listata e prazna");

}

**return** first;

}

**public** **void** insertFirst(E o) {

DLLNode<E> ins = **new** DLLNode<E>(o, **null**, first);

**if** (first == **null**)

last = ins;

**else**

first.pred = ins;

first = ins;

}

**public** **void** insertLast(E o) {

**if** (first == **null**)

insertFirst(o);

**else** {

DLLNode<E> ins = **new** DLLNode<E>(o, last, **null**);

last.succ = ins;

last = ins;

}

}

**public** **void** insertAfter(E o, DLLNode<E> after) {

**if**(after==last){

insertLast(o);

**return**;

}

DLLNode<E> ins = **new** DLLNode<E>(o, after, after.succ);

after.succ.pred = ins;

after.succ = ins;

}

**public** **void** insertBefore(E o, DLLNode<E> before) {

**if**(before == first){

insertFirst(o);

**return**;

}

DLLNode<E> ins = **new** DLLNode<E>(o, before.pred, before);

before.pred.succ = ins;

before.pred = ins;

}

**public** E deleteFirst() {

**if** (first != **null**) {

DLLNode<E> tmp = first;

first = first.succ;

**if** (first != **null**) first.pred = **null**;

**if** (first == **null**)

last = **null**;

**return** tmp.element;

} **else**

**return** **null**;

}

**public** E deleteLast() {

**if** (first != **null**) {

**if** (first.succ == **null**)

**return** deleteFirst();

**else** {

DLLNode<E> tmp = last;

last = last.pred;

last.succ = **null**;

**return** tmp.element;

}

}

// else throw Exception

**return** **null**;

}

**public** E delete(DLLNode<E> node) {

**if**(node==first){

deleteFirst();

**return** node.element;

}

**if**(node==last){

deleteLast();

**return** node.element;

}

node.pred.succ = node.succ;

node.succ.pred = node.pred;

**return** node.element;

}

@Override

**public** String toString() {

String ret = **new** String();

**if** (first != **null**) {

DLLNode<E> tmp = first;

ret += tmp + " ";

**while** (tmp.succ != **null**) {

tmp = tmp.succ;

ret += tmp + " ";

}

} **else**

ret = "Prazna lista!!!";

**return** ret;

}

**public** String toStringR() {

String ret = **new** String();

**if** (last != **null**) {

DLLNode<E> tmp = last;

ret += tmp + " ";

**while** (tmp.pred != **null**) {

tmp = tmp.pred;

ret += tmp + " ";

}

} **else**

ret = "Prazna lista!!!";

**return** ret;

}

**public** DLLNode<E> getFirst() {

**return** first;

}

**public** DLLNode<E> getLast() {

**return** last;

}

**public** **void** izvadiDupliIPrebroj(){

}

}

**public** **class** DLLRotateLeft {

**public** **static** **void** DLLRotateLeft(DLL<Integer> list, int k) {

}

**public** **static** **void** main(String[] args) **throws** IOException {

// **TODO** Auto-generated method stub

BufferedReader br = **new** BufferedReader(**new** InputStreamReader(System.***in***));

String s;

String[] pom;

**int** i, n, k, broj;

DLL<Integer> list = **new** DLL<Integer>();

s = br.readLine();

n = Integer.*parseInt*(s);

s = br.readLine();

pom = s.split(" ");

**for** (i = 0; i < n; i++) {

broj = Integer.*parseInt*(pom[i]);

list.insertLast(broj);

}

s = br.readLine();

k = Integer.*parseInt*(s);

DLL*RotateLeft*(list,k);

System.***out***.println(list);

}

}