**2. Спој листи - Java Задача 1 (0 / 0)**

Дадени се две еднострано поврзани листи чии јазли содржат по еден природен број. Листите се сортирани во растечки редослед. Треба да се спојат двете листи во една така што резултантната листа да е сортирана. Сортирањето е подредување со слевање. Јазлите кои се јавуваат како дупликати (од иста листа или од различна) да се отстранат. Во првиот ред од влезот е даден бројот на јазли во првата листа, потоа во вториот ред се дадени броевите од кои се составени јазлите по редослед во првата листа, па во третиот ред е даден бројот на јазли во втората листа, и на крај во четвртиот ред броевите од кои се составени јазлите по редослед во втората листа. На излез треба да се испечатат јазлите по редослед во резултантната споена листа. Име на класата: SLLJoinLists

Вашето решение:

Програмски јазик: Java

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.Iterator;

import java.util.NoSuchElementException;

class SLLNode<E extends Comparable<E>> {

protected E element;

protected SLLNode<E> succ;

public SLLNode(E elem, SLLNode<E> succ) {

this.element = elem;

this.succ = succ;

}

@Override

public String toString() {

return element.toString();

}

}

class SLL<E extends Comparable<E>> {

private SLLNode<E> first;

public SLL() {

// Construct an empty SLL

this.first = null;

}

public void deleteList() {

first = null;

}

public int length() {

int ret;

if (first != null) {

SLLNode<E> tmp = first;

ret = 1;

while (tmp.succ != null) {

tmp = tmp.succ;

ret++;

}

return ret;

} else

return 0;

}

@Override

public String toString() {

String ret = new String();

if (first != null) {

SLLNode<E> tmp = first;

ret += tmp + "->";

while (tmp.succ != null) {

tmp = tmp.succ;

ret += tmp + "->";

}

} else

ret = "Prazna lista!!!";

return ret;

}

public void insertFirst(E o) {

SLLNode<E> ins = new SLLNode<E>(o, first);

first = ins;

}

public void insertAfter(E o, SLLNode<E> node) {

if (node != null) {

SLLNode<E> ins = new SLLNode<E>(o, node.succ);

node.succ = ins;

} else {

System.out.println("Dadenot jazol e null");

}

}

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

if (first != null) {

SLLNode<E> tmp = first;

if (first == before) {

this.insertFirst(o);

return;

}

// ako first!=before

while (tmp.succ != before)

tmp = tmp.succ;

if (tmp.succ == before) {

SLLNode<E> ins = new SLLNode<E>(o, before);

tmp.succ = ins;

} else {

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

}

} else {

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

}

}

public void insertLast(E o) {

if (first != null) {

SLLNode<E> tmp = first;

while (tmp.succ != null)

tmp = tmp.succ;

SLLNode<E> ins = new SLLNode<E>(o, null);

tmp.succ = ins;

} else {

insertFirst(o);

}

}

public SLLNode<E> getFirst() {

return first;

}

public Iterator<E> iterator() {

// Return an iterator that visits all elements of this list, in

// left-to-right order.

return new LRIterator<E>();

}

// //////////Inner class ////////////

private class LRIterator<E extends Comparable<E>> implements Iterator<E> {

private SLLNode<E> place;

@SuppressWarnings("unchecked")

private LRIterator() {

place = (SLLNode<E>) first;

}

public boolean hasNext() {

return (place != null);

}

public E next() {

if (place == null)

throw new NoSuchElementException();

E nextElem = place.element;

place = place.succ;

return nextElem;

}

public void remove() {

// Not implemented

}

}

public SLL<E> joinLists(SLL<E> in) {

//Vashiot kod tuka...

}

}

public class SLLJoinLists {

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

SLL<Integer> lista1 = new SLL<Integer>(), lista2 = new SLL<Integer>();

BufferedReader stdin = new BufferedReader(new InputStreamReader(

System.in));

String s = stdin.readLine();

int N = Integer.parseInt(s);

s = stdin.readLine();

String[] pomniza = s.split(" ");

for (int i = 0; i < N; i++) {

lista1.insertLast(Integer.parseInt(pomniza[i]));

}

s = stdin.readLine();

N = Integer.parseInt(s);

s = stdin.readLine();

pomniza = s.split(" ");

for (int i = 0; i < N; i++) {

lista2.insertLast(Integer.parseInt(pomniza[i]));

}

SLL<Integer> spoeni = lista1.joinLists(lista2);

Iterator<Integer> it = spoeni.iterator();

while (it.hasNext()) {

System.out.print(it.next());

if(it.hasNext())

System.out.print(" ");

}

}

}