DLLPrvKolokvium2020Termin4

Да се направи метода која „нанижува“ (надоврзува) две двострано поврзани листи. За дадени две листи, втората листа треба да се слепи на првата, т.е. резултатот да биде една двострано поврзана листа којашто ги содржи прво јазлите од првата листа, па јазлите од втората листа. На крајот на нанижаната листа да се додаде ново теме, во коешто се содржи сумата на сите вредности од двете дадени листи.

**Влез**: Во првиот ред е дадени се два броја: M и N, што ги означуваат должините на листите. Во вториот ред се дадени М броеви - елементите на првата листа. Во третиот ред дадени се N броеви - елементите на втората листа.

**Излез**: Броевите од финалната листа. Во првиот ред се печатат по редослед од прв до последен, а во вториот обратно, од последен до прв.

**Забелешка**: Даден е основниот код од задачата со имплементиран влез и излез како и структурата која треба да се користи. Дадена е и класата  Nanizuvanje, во која  на точно означеното место ќе трeба да се имплементират бараните услови. Вашиот код треба да биде напишан само таму, без измени во останатиот дел од кодот.

Пример

5

5

1 3 4 6 7

9 8 5 2 1

1 3 4 6 7 9 8 5 2 1 46

46 1 2 5 8 9 7 6 4 3 1

**import** java.util.Scanner;

**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** APSKol {

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

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

Scanner input = **new** Scanner(System.***in***);

//citanje na broj na elementi

**int** m = input.nextInt();

**int** n = input.nextInt();

DLL<Integer> lista1 = **new** DLL<Integer>();

DLL<Integer> lista2 = **new** DLL<Integer>();

//citaj listi, edna po edna

**for** (**int** i = 0; i < m; i++) {

**int** el = input.nextInt();

lista1.insertLast(el);

}

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

**int** el = input.nextInt();

lista2.insertLast(el);

}

DLLNode<Integer> current = lista1.getFirst();

//vasiot kod tuka..

//pecatenje

System.***out***.println(lista1.toString());

System.***out***.println(lista1.toStringR());

}

}