Prvi test u drugoj godini (prelazak na javu)

1. Napisati program koji u datom izrazu, koji se sastoji od otvorenih i zatvorenih malih zagrada, brojem

0 menja otvorene zagrade koje su pravilno zatvorene, brojem 1 menja zatvorene zagrade za koje

postoji odgovarajuća otvorena zagrada, a brojem -1 nepravilno postavljene otvorene i zatvorene

zagrade.

Napomena: Koristiti stek kao strukturu podataka. Dozvoljeno je koristiti pomoćni stek.

Primeri:

ULAZ: ULAZ: ULAZ:

(() ()) ((())(()))))

IZLAZ: IZLAZ: IZLAZ:

-101 01-1 0001100111-1-1

2. Napisati program koji učitava dva prirodna broja M i N kao niz cifara, a zatim štampa njihov zbir.

Napomena: Koristiti stek kao strukturu podataka.

Primeri:

ULAZ: ULAZ:

123456789123456789 89321809894321098894

111111111111111111 54892489123098089234

IZLAZ: IZLAZ:

234567900234567900 144214299017419188128

3. Napisati program, koristeći metode reda, koji stvara novi red koji sadrži samo negativne elemente iz

zadatog reda. Početni red mora ostati očuvan. Uputstvo: koristiti pomoćni red.

Ispisati oba reda.

Odradjena sva tri zadatka:

import java.util.LinkedList;

import java.util.Queue;

import java.util.Scanner;

// Press Shift twice to open the Search Everywhere dialog and type `show whitespaces`,

// then press Enter. You can now see whitespace characters in your code.

//TRECI ZADATAK

public class Main {

public static void main(String[] args) {

Scanner citac=new Scanner(System.in);

Queue <Integer> pocetnired=new LinkedList<Integer>();

Queue <Integer> pocetniredkopija=new LinkedList<Integer>();

Queue <Integer> red=new LinkedList<Integer>();

int pocetna=citac.nextInt();

while(pocetna!=0000)

{

pocetnired.add(pocetna);

pocetna=citac.nextInt();

}

pocetniredkopija.addAll(pocetnired);

while (pocetniredkopija.size()!=0)

{

if(pocetniredkopija.peek()<0)

{

red.add(pocetniredkopija.peek());

pocetniredkopija.poll();

}

else pocetniredkopija.poll();

}

System.out.println("Pocetni red: "+pocetnired);

System.out.println("Novi red: "+red);

}

}

import java.util.Scanner;

import java.util.Stack;

**//DRUGI ZADATAK**

public class Main {

public static void main(String[] args) {

Scanner citac=new Scanner(System.in);

Stack<Integer> prvistek = new Stack<Integer>();

Stack<Integer> drugistek = new Stack<Integer>();

Stack<Integer> rezstek = new Stack<Integer>();

int ostatak=0;

int ukupno;

String prvibr= citac.next();

String drugibr= citac.next();

for (int i = 0; i <prvibr.length() ; i++)

{

prvistek.push(prvibr.charAt(i)-48);

}

for (int i = 0; i <drugibr.length() ; i++)

{

drugistek.push(drugibr.charAt(i)-48);

}

while(prvistek.size()!=0&&drugistek.size()!=0)

{

ukupno=prvistek.pop()+drugistek.pop()+ostatak;

rezstek.push(ukupno%10);

ostatak=ukupno/10;

}

while(prvistek.size()!=0)

{

ukupno=prvistek.pop()+ostatak;

rezstek.push(ukupno%10);

ostatak=ukupno/10;

}

while(drugistek.size()!=0)

{

ukupno=drugistek.pop()+ostatak;

rezstek.push(ukupno%10);

ostatak=ukupno/10;

}

if (ostatak!=0)

{

rezstek.push(ostatak);

}

while(rezstek.size()!=0)

{

System.out.print(rezstek.pop());

}

}

}

import java.util.LinkedList;

import java.util.Queue;

import java.util.Scanner;

import java.util.Stack;

//PRVI ZADATAK

public class Main {

public static void main(String[] args) {

Scanner citac=new Scanner(System.in);

Stack<Character> stek=new Stack<>();

Stack<Character> stek2=new Stack<>();

String niska=citac.next();

for (int i=0;i<niska.length();i++){

if(!stek.isEmpty())

{

if(niska.charAt(i)==stek.peek())

{

stek.pop();

}

else{stek.add(niska.charAt(i));}

}

else stek.add(niska.charAt(i));

}

while (!stek.isEmpty())

{

stek2.add(stek.peek());

stek.pop();

}

while (!stek2.isEmpty()){

System.out.print(stek2.peek());

stek2.pop();

}

}

}

Prvi pismeni u drugoj godini

1. Napisati program koji učitava niz celih brojeva A dužine N, a zatim za svaki element niza

štampa prvi veći element koji se nalazi posle njega u nizu. Za elemente za koje ne postoji

takav element, štampati -1.

Napomena: Rešenje koje koristi dve ugnježdene for petlje i ostala rešenje slične

složenosti neće biti priznata. Potrebno je koristiti stek kao strukturu podataka.

Primeri:

ULAZ: ULAZ:

4 4

4 5 2 25 13 7 6 12

IZLAZ: IZLAZ:

5 25 25 -1 -1 12 12 -1

2. Za uneto binarno stablo pretrage od strane korisnika, prebrojte ukupan broj čvorova koji

se nalaze unutar nekog opsega. Korisnik unosi koliko ima čvorova, vrednosti u

čvorovima, najmanji i najveći broj opsega pri čemu je sigurno a&lt;b.

Na primer, ukupan broj čvorova u opsegu [12, 20] u sledećem BST je 4. Čvorovi su 12,

15, 18 i 20.

3. Napisati program kojim se gradi binarno stablo pretrage. Broj čvorova stabla n unosi se

na početku programa. U čvorove binarnog stabla unosi se sledećih n celih brojeva sa

konzole.

• Ispisati broj čvorova binarnog stabla koji imaju oba deteta, jedno dete i broj

listova.

• Izbrisati čvor sa najvećom vrednošću u stablu.

• Ispisati elemente stabla u rastućem poretku.

4. Napišite Java program koji vodi evidenciju studenata na fakultetu koristeći Heš tabelu.

Program treba da podržava sledeće funkcionalnosti:

1. Dodavanje novog studenta:

 Omogućite unos podataka o novom studentu (ime, prezime, broj indeksa,

godina studija).

 Svaki student treba da ima jedinstveni broj indeksa.

2. Prikaz svih studenata:

 Prikazuje sve dodate studente sa svim njihovim informacijama.

3. Pretraga studenta po broju indeksa:

 Omogućite korisniku da unese broj indeksa i program treba da prikaže

podatke o studentu sa tim brojem indeksa, ako student postoji.

Primer izgleda rada programa nakon pokretanja:

Dobrodošli u evidenciju studenata na fakultetu!

Unesite broj akcije:

1. Dodavanje novog studenta

2. Prikaz svih studenata

3. Pretraga studenta po broju indeksa

0. Izlaz

Vaš izbor: 1

Unesite ime studenta: Marko

Unesite prezime studenta: Petrović

Unesite broj indeksa studenta: 12345

Unesite godinu studija: 2

Student je uspešno dodat.

Unesite broj akcije:

1. Dodavanje novog studenta

2. Prikaz svih studenata

3. Pretraga studenta po broju indeksa

0. Izlaz

Vaš izbor: 2

Svi studenti na fakultetu:

Broj indeksa: 12345, Ime: Marko, Prezime: Petrović, Godina studija: 2

Unesite broj akcije:

1. Dodavanje novog studenta

2. Prikaz svih studenata

3. Pretraga studenta po broju indeksa

0. Izlaz

Vaš izbor: 3

Unesite broj indeksa studenta za pretragu: 12345

Student sa brojem indeksa 12345:

Ime: Marko

Prezime: Petrović

Godina studija: 2

Unesite broj akcije:

1. Dodavanje novog studenta

2. Prikaz svih studenata

3. Pretraga studenta po broju indeksa

0. Izlaz

Vaš izbor: 0

Program je završen. Hvala!

**////////Drugi zadatak**

import java.util.Scanner;

class BST\_class {

//node class that defines BST node

class Node {

int key;

Node left, right;

public Node(int data){

key = data;

left = right = null;

}

}

// BST root node

Node root;

// Constructor for BST =>initial empty tree

BST\_class(){

root = null;

}

//delete a node from BST

void deleteKey(int key) {

root = delete\_Recursive(root, key);

}

//recursive delete function

Node delete\_Recursive(Node root, int key) {

//tree is empty

if (root == null) return root;

//traverse the tree

if (key < root.key) //traverse left subtree

root.left = delete\_Recursive(root.left, key);

else if (key > root.key) //traverse right subtree

root.right = delete\_Recursive(root.right, key);

else {

// node contains only one child

if (root.left == null)

return root.right;

else if (root.right == null)

return root.left;

// node has two children;

//get inorder successor (min value in the right subtree)

root.key = minValue(root.right);

// Delete the inorder successor

root.right = delete\_Recursive(root.right, root.key);

}

return root;

}

int minValue(Node root) {

//initially minval = root

int minval = root.key;

//find minval

while (root.left != null) {

minval = root.left.key;

root = root.left;

}

return minval;

}

// insert a node in BST

void insert(int key) {

root = insert\_Recursive(root, key);

}

//recursive insert function

Node insert\_Recursive(Node root, int key) {

//tree is empty

if (root == null) {

root = new Node(key);

return root;

}

//traverse the tree

if (key < root.key) //insert in the left subtree

root.left = insert\_Recursive(root.left, key);

else if (key > root.key) //insert in the right subtree

root.right = insert\_Recursive(root.right, key);

// return pointer

return root;

}

// method for inorder traversal of BST

void inorder() {

inorder\_Recursive(root);

}

// recursively traverse the BST

void inorder\_Recursive(Node root) {

if (root != null) {

inorder\_Recursive(root.left);

System.out.print(root.key + " ");

inorder\_Recursive(root.right);

}

}

// method for postorder traversal of BST

void postOrder() {

postOrder\_Recursive(root);

}

//PostOrder Traversal - Left:Right:rootNode (LRn)

void postOrder\_Recursive(Node root) {

if (root == null)

return;

// first traverse left subtree recursively

postOrder\_Recursive(root.left);

// then traverse right subtree recursively

postOrder\_Recursive(root.right);

// now process root node

System.out.print(root.key + " ");

}

// method for postorder traversal of BST

void preOrder() {

preOrder\_Recursive(root);

}

//PreOrder Traversal - rootNode:Left:Right (nLR)

void preOrder\_Recursive(Node root) {

if (root == null)

return;

//first print root node first

System.out.print(root.key + " ");

// then traverse left subtree recursively

preOrder\_Recursive(root.left);

// next traverse right subtree recursively

preOrder\_Recursive(root.right);

}

boolean search(int key) {

root = search\_Recursive(root, key);

if (root!= null)

return true;

else

return false;

}

//recursive insert function

Node search\_Recursive(Node root, int key) {

// Base Cases: root is null or key is present at root

if (root==null || root.key==key)

return root;

// val is greater than root's key

if (root.key > key)

return search\_Recursive(root.left, key);

// val is less than root's key

return search\_Recursive(root.right, key);

}

int brojListova() {

if ( root == null )

return 0;

return dohvatiBrojListova(root);

}

int dohvatiBrojListova( Node root ) {

int brListovaLokalno = 0;

if ( root.left == null && root.right == null )

return 1; // U pitanju je list, na zbir dodaj 1. Nakon ovog return-a sigurno se vracamo na mesto (\*) ili (\*\*)

else {

if ( root.left != null )

brListovaLokalno += dohvatiBrojListova( root.left ); // MESTO (\*)

if ( root.right != null )

brListovaLokalno += dohvatiBrojListova( root.right ); // MESTO (\*\*)

return brListovaLokalno;

}

}

int opseg(int donjagranica,int gornjagranica)

{

return upotrebiopseg(root,donjagranica,gornjagranica);

}

int upotrebiopseg(Node node, int donjagranica, int gornjagranica)

{

if(node==null)return 0;

else if(node.key>=donjagranica&&node.key<=gornjagranica) return 1+upotrebiopseg(node.left,donjagranica,gornjagranica)+upotrebiopseg(node.right,donjagranica,gornjagranica);

else if(node.key<donjagranica) return upotrebiopseg(node.right, donjagranica, gornjagranica);

else return upotrebiopseg(node.left,donjagranica,gornjagranica);

}

int jednodete()

{

return upotrebijednodete(root);

}

int upotrebijednodete(Node node)

{

int brojac=0;

if(node==null)return 0;

if(node.left!=null^node.right!=null) brojac++;

return brojac+upotrebijednodete(node.left)+upotrebijednodete(node.right);

}

int dvadeteta()

{

return upotrebidvadeteta(root);

}

int upotrebidvadeteta(Node node)

{

int zbir=0;

if(node==null)return 0;

if(node.left!=null&&node.right!=null) zbir++;

return zbir+upotrebidvadeteta(node.left)+upotrebidvadeteta(node.right);

}

int maxValue()

{

return(upotrebamaxValue(root));

}

int upotrebamaxValue(Node root) {

if(root.right!=null)

return upotrebamaxValue(root.right);

else return root.key;

}

}

class Main{

public static void main(String[] args) {

//create a BST object

BST\_class bst = new BST\_class();

Scanner citac=new Scanner(System.in);

System.out.println("Koliko clanova ima");

int n=citac.nextInt();

System.out.println("unesite clanove");

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

bst.insert(citac.nextInt());

}

System.out.println("unesite donju granicu");

int donjagranica=citac.nextInt();

System.out.println("unesite gornju granicu");

int gornjagranica=citac.nextInt();

System.out.println(bst.opseg(donjagranica,gornjagranica));

System.out.println(bst.dvadeteta());

System.out.println(bst.jednodete());

System.out.println(bst.brojListova());

bst.deleteKey(bst.maxValue());

bst.inorder();

System.out.println(n);

bst.inorder();

}

}

**////Prvi zadatak**

import java.util.\*;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int N = scanner.nextInt();

int[] niz = new int[N];

Stack<Integer> stack = new Stack<Integer>();

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

niz[i] = scanner.nextInt();

}

int[] niz2 = new int[N];

for (int i = N - 1; i >= 0; i--) {

if (stack.isEmpty()) {

stack.push(niz[i]);

niz2[i] = -1;

} else {

if (stack.peek() > niz[i]) {

niz2[i] = stack.peek();

stack.push(niz[i]);

} else {

stack.pop();

i++;

}

}

}

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

System.out.println(niz2[i]);

}

}

}

**////Treci zadatak**

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

class Student{

String ime;

String prezime;

int brindeksa;

int godinastudija;

public Student(String ime, String prezime, int brindeksa, int godinastudija) {

this.ime = ime;

this.prezime = prezime;

this.brindeksa = brindeksa;

this.godinastudija = godinastudija;

}

public String getIme() {

return ime;

}

public String getPrezime() {

return prezime;

}

public int getBrindeksa() {

return brindeksa;

}

public int getGodinastudija() {

return godinastudija;

}

@Override

public String toString() {

return "Broj indeksa: "+ brindeksa + ","+ "Ime:"+ ime+", "+"Prezime:"+prezime+","+"Godina studija:"+ godinastudija;

}

}

class Main{

public static void main(String[] args) {

Map<Integer,Student> studenti=new HashMap<>();

Scanner skener=new Scanner(System.in);

System.out.println("Dobrodošli u evidenciju studenata na fakultetu!");

System.out.println("Unesite broj akcije:");

System.out.println("1. Dodavanje novog studenta");

System.out.println("2. Prikaz svih studenata");

System.out.println("3. Pretraga studenta po broju indeksa");

System.out.println("0. Izlaz");

int izbor=skener.nextInt();

while(izbor!=0)

{

if(izbor==1)

{

System.out.println("Vaš izbor: 1");

System.out.println("unesite ime studenta");

String ime=skener.next();

System.out.println("unesite prezime studenta");

String prezime=skener.next();

System.out.println("unesite broj indeksa studenta");

int brindeksa=skener.nextInt();

System.out.println("unesite godinu studija");

int godinastudija=skener.nextInt();

Student student=new Student(ime,prezime,brindeksa,godinastudija);

studenti.put(brindeksa,student);

System.out.println("Student je uspešno dodat.");

}

else if (izbor==2)

{

System.out.println("Svi studenti na fakultetu:");

for (int x:studenti.keySet())

{

System.out.println(studenti.get(x));

}

}

else if(izbor==3)

{

System.out.println("Unesite broj indeksa studenta za pretragu:");

int brindeksa=skener.nextInt();

System.out.println(studenti.get(brindeksa));

}

System.out.println("Unesite vas izbor ponovo");

izbor=skener.nextInt();

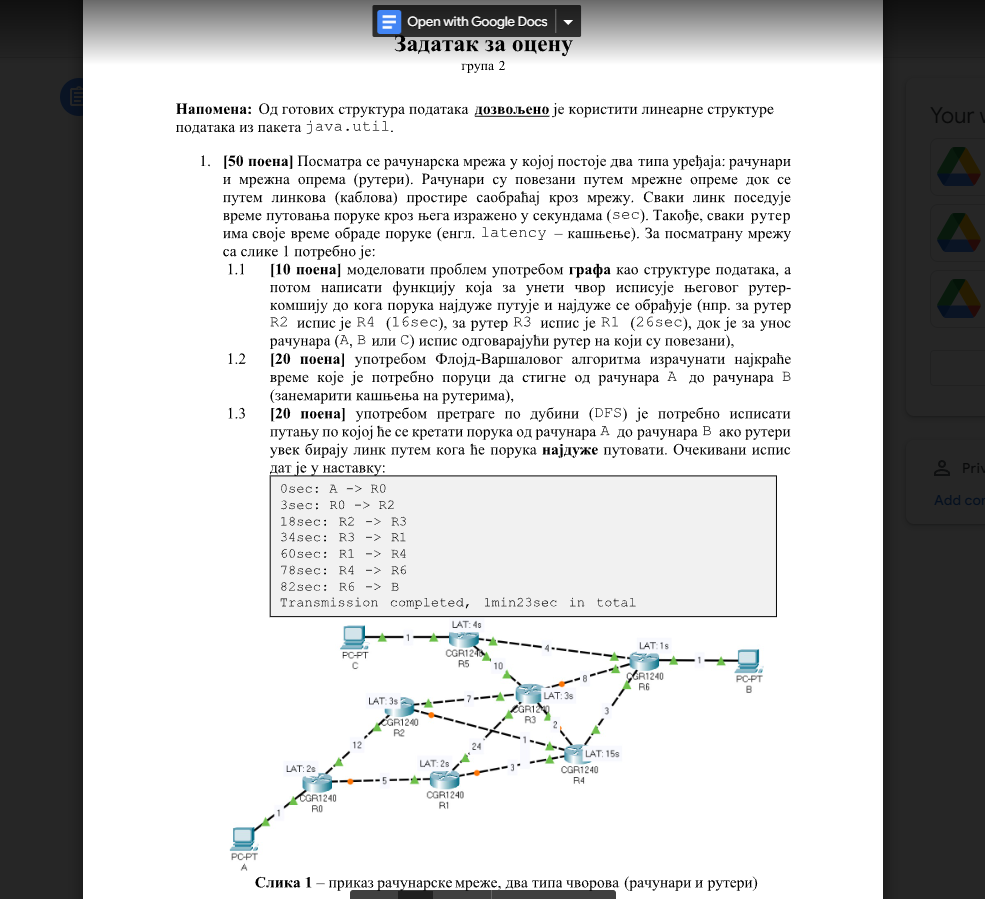
}

System.out.println("Program je završen. Hvala!");

}

}

**Drugi test u drugoj godini**



**Potrebni fajlovi sa pokretanje zadatka**

1. **Cvor.java**
2. **Graf.java**
3. **Main.java**

**1.**

public class Cvor {

private String ime;

private boolean isPC;

private int latency;

private int index;

public Cvor(String r1, boolean b, int i) {

}

public Cvor(String ime, boolean isPC, int latency, int index) {

this.ime = ime;

this.isPC = isPC;

this.latency = latency;

this.index = index;

}

public String getIme() {

return ime;

}

public void setIme(String ime) {

this.ime = ime;

}

public boolean isPC() {

return isPC;

}

public void setPC(boolean PC) {

isPC = PC;

}

public int getLatency() {

return latency;

}

public void setLatency(int latency) {

this.latency = latency;

}

public int getIndex() {

return index;

}

public void setIndex(int index) {

this.index = index;

}

}

**2.**

import java.util.Stack;

public class Graf {

int[][] matrica;

int dimenzija;

int trenutni=0;

Cvor[] niz;

public void postaviDimenziju(int i) {

dimenzija=i;

niz=new Cvor[dimenzija];

matrica=new int [dimenzija][dimenzija];

for (int j = 0; j <dimenzija ; j++) {

for (int k = 0; k <dimenzija ; k++) {

if(k==j)matrica[j][k]=0;

else matrica[j][k]=Integer.MAX\_VALUE/2;

}

}

}

public void dodajCvor(String r1, boolean b, int i) {

niz[trenutni]=new Cvor(r1,b,i, trenutni);

trenutni++;

}

public void dodajLink(String a, String r0, int i) {

int x=-1,y=-1;

for (int j = 0; j < dimenzija ; j++) {

if (niz[j].getIme()==a)x=j;

if(niz[j].getIme()==r0)y=j;

}

matrica[x][y]=i;

}

public Cvor najdaljiKomsija(String r1) {

int maxveza=0;

int maxcvor=0;

int index=-1;

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

if(niz[i].getIme()==r1)index=i;

}

for (int j = 0; j < dimenzija; j++) {

if((matrica[index][j]+niz[j].getLatency())>maxveza&&matrica[index][j]!=Integer.MAX\_VALUE/2)

{

maxveza=matrica[index][j]+niz[j].getLatency();

maxcvor=j;

}

}

return niz[maxcvor];

}

public String najkraceVreme(String a, String b) {

for(int k=0; k<dimenzija; k++){

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

for(int j=0; j<dimenzija; j++){

if (matrica[i][j] > matrica[i][k] + matrica[k][j]){

matrica[i][j] = matrica[i][k] + matrica[k][j];

}

}

}

}

int x=-1,y=-1;

for (int j = 0; j < dimenzija ; j++) {

if (niz[j].getIme()==a)x=j;

if(niz[j].getIme()==b)y=j;

}

return (""+matrica[x][y]);

}

public void testiranjeSaobracajaOdRacunaraA() {

Stack<Integer> s=new Stack<>();

Stack<Integer> poseceni=new Stack<>();

Stack<Integer> resenja=new Stack<>();

int duzagrana=Integer.MIN\_VALUE;

int trenutni=-1;

s.push(0);

while(s.peek()!=6)

{

for (int j= 0; j < 7 ; j++)

{

if (matrica[s.peek()][j]+niz[j].getLatency() > duzagrana && matrica[s.peek()][j] != Integer.MAX\_VALUE/2&&!poseceni.contains(j))

{

duzagrana = matrica[s.peek()][j]+niz[j].getLatency(); //2

trenutni = j; //3

}

}

System.out.println(duzagrana);

System.out.println(trenutni);

poseceni.push(trenutni);

duzagrana=Integer.MIN\_VALUE;

s.pop();

s.push(trenutni);

}

}

}

**3.**

import java.util.\*;

public class Main {

public static void prvizadatak(int daticvor, int[][] graf)

{

int maxveza=0;

int maxcvor=0;

for (int i = 0; i < 7; i++) {

for (int j = 0; j < 7; j++) {

if(graf[i][j]>maxveza&&i==daticvor)

{

maxveza=graf[i][j];

maxcvor=j;

}

}

}

System.out.printf(" Max komsa za cvor "+daticvor+" je "+maxcvor);

}

public static void drugizadatak(int[][] graf)

{

Stack<Integer> s=new Stack<>();

Stack<Integer> poseceni=new Stack<>();

s.push(0);

int i=0;

int brcvorova=0;

int kracagrana=Integer.MAX\_VALUE;

int trenutni=-1;

while(s.peek()!=5)

{

if(s.peek()==i)

{

for (int j= 0; j < 7 ; j++)

{

if (graf[s.peek()][j] < kracagrana && graf[s.peek()][j] != 0&&!poseceni.contains(j))

{

kracagrana = graf[s.peek()][j]; //5

trenutni = j; //2

System.out.println(kracagrana);

System.out.println(trenutni);

}

}

poseceni.push(trenutni);

kracagrana=Integer.MAX\_VALUE;

brcvorova++;

s.pop();

s.push(trenutni);

trenutni=-1;

}

i++;

}

System.out.printf("broj rutera je "+brcvorova);

}

public static void main(String[] args) { Graf g = new Graf(); g.postaviDimenziju(10); g.dodajCvor("R0", false, 2);

g.dodajCvor("R1", false, 2);

g.dodajCvor("R2", false, 3);

g.dodajCvor("R3", false, 3);

g.dodajCvor("R4", false, 15);

g.dodajCvor("R5", false, 4);

g.dodajCvor("R6", false, 1);

g.dodajCvor("A", true, 0);

g.dodajCvor("B", true, 0);

g.dodajCvor("C", true, 0);

g.dodajLink("A", "R0", 1);

g.dodajLink("B", "R6", 1);

g.dodajLink("C", "R5", 1);

g.dodajLink("R0", "A", 1);

g.dodajLink("R0", "R1", 5);

g.dodajLink("R0", "R2", 12);

g.dodajLink("R1", "R3", 24);

g.dodajLink("R1", "R4", 3);

g.dodajLink("R2", "R0", 12);

g.dodajLink("R2", "R4", 1);

g.dodajLink("R2", "R3", 7);

g.dodajLink("R3", "R1", 24);

g.dodajLink("R3", "R5", 10);

g.dodajLink("R3", "R6", 8);

g.dodajLink("R4", "R6", 3);

g.dodajLink("R4", "R3", 2);

g.dodajLink("R5", "C", 1);

g.dodajLink("R5", "R3", 10);

g.dodajLink("R5", "R6", 4);

g.dodajLink("R6", "B", 1);

g.dodajLink("R6", "R4", 3);

g.dodajLink("R6", "R5", 4);

System.out.println("Najdalji komsija od R1 je " + g.najdaljiKomsija("R1").getIme()); System.out.println("Najdalji komsija od R2 je " + g.najdaljiKomsija("R2").getIme()); System.out.println("Najdalji komsija od R3 je " + g.najdaljiKomsija("R3").getIme());

System.out.println("Najkrace vreme A-B je " + g.najkraceVreme("A", "B"));

g.testiranjeSaobracajaOdRacunaraA(); // tacka 1.3

}

}