Posted by: Nermin Milišić - Monday, April 18, 2016 7:00:35 PM

Originally Posted by: Denis Mušić

```
#include
using namespace std;
void pomocnaFunkcija(char* pok1, char* pok2) {
char znak = *pok2;
*pok2 = *pok1;
*pok1 = znak;
}
void rekurzija(char* pok, int pocetak, int kraj) {
if (kraj - pocetak == 1)
cout << pok << endl;
else {
for (int i = 0; i < kraj - pocetak; i++) {
pomocnaFunkcija(&pok[pocetak], &pok[pocetak + i]);
rekurzija(pok, pocetak + 1, kraj);
pomocnaFunkcija(&pok[pocetak], &pok[pocetak + i]);
}
}
}
void main() {
char tekst[] = "fit";
rekurzija(tekst, 0, strlen(tekst));
}
```

Ako nekom nece da iskompajlira ovaj zadatak treba includati <string.h> biblioteku. lp

Posted by: Denis Mušić - Saturday, February 20, 2016 10:58:51 AM

```
#include <iostream>
using namespace std;
void pomocnaFunkcija(char* pok1, char* pok2) {
char znak = *pok2;
*pok2 = *pok1;
*pok1 = znak;
}
void rekurzija(char* pok, int pocetak, int kraj) {
if (kraj - pocetak == 1)
cout << pok << endl;
else {
for (int i = 0; i < kraj - pocetak; i++) {
pomocnaFunkcija(&pok[pocetak], &pok[pocetak + i]);
rekurzija(pok, pocetak + 1, kraj);
pomocnaFunkcija(&pok[pocetak], &pok[pocetak + i]);
}
}
}
void main() {
char tekst[] = "fit";
rekurzija(tekst, 0, strlen(tekst));
}
```

Posted by: Denis Mušić - Saturday, February 20, 2016 10:58:14 AM

```
#include<iostream>
using namespace std;
struct Datum {
int * _dan, *_mjesec, *_godina;
void Unos(int d, int m, int g) {
_dan = new int(d);
_mjesec = new int(m);
_godina = new int(g);
}
void Ispis(){cout << *_dan << "/" << *_mjesec << "/" << *_godina << endl; }
void Dealociraj() {
delete _dan; delete _mjesec; delete _godina;
_dan = _mjesec = _godina = nullptr;
```

```
}
};
enum VrstaKursa{HtmlCSSJavaScript, SoftwareEngeneeringFundamentals, MasteringSQL,
WindowsSecurity };
struct Kurs {
VrstaKursa _vrsta;
Datum * _pocetak;
Datum * _kraj;
char * _imePredavaca;
void Unos(VrstaKursa vrsta, Datum pocetak, Datum kraj, char * predavac) {
_vrsta = vrsta;
_pocetak = new Datum();
_pocetak->Unos(*pocetak._dan, *pocetak._mjesec, *pocetak._godina);
_kraj = new Datum();
_kraj->Unos(*kraj._dan, *kraj._mjesec, *kraj._godina);
_imePredavaca = new char[strlen(predavac) + 1];
strcpy_s(_imePredavaca, strlen(predavac) + 1, predavac);
}
void Ispis() {
cout << _vrsta << " (" << _imePredavaca << ")" << endl;
cout << _pocetak->Ispis()<< " " << _kraj->Ispis() << endl;
}
void Dealociraj() {
_pocetak->Dealociraj(); delete _pocetak; _pocetak = nullptr;
_kraj->Dealociraj(); delete _kraj; _kraj = nullptr;
delete[] _imePredavaca; _imePredavaca = nullptr;
}
};
struct Kandidat {
char * _imePrezime;
Kurs * _pohadjaniKursevi;
float * _ostvareniUspjeh;
int _tretnutnoKurseva;
void Unos(char * imePrezime) {
_imePrezime = new char[strlen(imePrezime) + 1];
strcpy_s(_imePrezime, strlen(imePrezime)+1, imePrezime);
_pohadjaniKursevi = nullptr;
_ostvareniUspjeh = nullptr;
_tretnutnoKurseva = 0;
}
void Dealociraj() {
delete[]_imePrezime; _imePrezime = nullptr;
for (int i = 0; i < _tretnutnoKurseva; i++)
_pohadjaniKursevi[i].Dealociraj();
delete[] _pohadjaniKursevi; _pohadjaniKursevi = nullptr;
delete[] _ostvareniUspjeh; _ostvareniUspjeh = nullptr;
}
void Ispis() {
cout << _imePrezime << endl;
for (size_t i = 0; i < _tretnutnoKurseva; i++){
_pohadjaniKursevi[i].Ispis();
cout << " Uspjeh: " << _ostvareniUspjeh[i] << endl;
}
}
bool DodajKurs(Kurs k) { /*dodaje podatke o novom kursu (nizove povećavati prilikom dodavanja svakog novog kursa). u jednom trenutku kandidat može prijaviti najviše 2 kursa. svaki kurs se okončava evidentiranjem ostvarenog uspjeha (0%-100%).ostvareni uspjeh se dodaje tek
nakon polaganja, pa neokončani kurs podrazumijeva onaj kurs kome nije evidentiran uspjeh. prilikom dodavanja kursa voditi računa da kandidat najviše dva puta prijaviti isti kurs, te nije moguće ponovo prijaviti kurs na kome je kandidat ostvario uspjeh veći od 60%. onemogućiti
dodavanja kurseva koji se održavaju u istom terminu kao neki od neokončanih kurseva (novi kurs se ne smije dodati dok u tom periodu postoji neokončani kurs). */
}
bool DodajUspjeh(Kurs * k, float uspjeh) {
/*dodaje uspjeh ostvaren na kursu k. da bi se uspjeh dodao kurs k treba postojati u listi pohađanih kurseva*/
}
int UkloniKurs(VrstaKursa vrsta) {
/*uklanja sve kurseve koji su vrste definisane parametrom. funkcija vraća broj uklonjenih kurseva. u slučaju uklanjanja kursa umanjiti veličinu niza.*/
}
};
Kandidat * Pretraga(Kandidat * k, int brojKandidata, char * predavac, Datum * OD, Datum * DO, float uspjeh) {
/*vraća listu kandidata koji su u određenom periodu (OD - DO) okončali kurs kod traženog predavača, te ostvarili uspjeh koji je veći od onog definisanog parametrom. iz liste isključiti one koji su kod traženog predavača poništavali uspjeh (ostvarili uspjeh manji od 60%) tj. ponovo
prijavljivali kurs (ne mora se raditi o istoj vrsti kursa)*/
}
void main() {
//izbjegavajte koristenje vlastitog imena i prezimena
//provjeriti validnost izvrsenja svih dostupnih funkcionalnosti
//nije potrebno da korisnik unosi vrijednosti parametara (možete ih predefinisati)
}
```

Posted by: Denis Mušić - Friday, January 22, 2016 11:00:43 AM

1. ZADATAK
Koristeći rekurziju napisati program koji će, Euklidovim postupkom, sumirati najveće zajedničke djelioce susjednih članova niza: {15, 10, 30, 21, 14}. Sumu najvećih zajedničkih djelilaca (nzd(15,10) + nzd(10,30) + …), koju vraća rekurzivna funkcija, ispisati u main-u.
2. ZADATAK
Koristeći prikazane strukture izvršiti definiciju funkcija na način koji odgovara opisu (komentarima) samih funkcija. Također, dozvoljeno je dati komentar na bilo koju liniju code-a koju bi trebalo unaprijediti ili da će eventualno uzrokovati grešku prilikom kompajliranja.

```
#include<iostream>
using namespace std;
struct Datum{
int * _dan, *_mjesec, *_godina;
void Unos(int d, int m, int g){
_dan = new int(d);
_mjesec = new int(m);
_godina = new int(g);
}
void Ispis(){cout << *_dan << "/" << *_mjesec << "/" << *_godina << endl;}
void Dealociraj(){
delete _dan; delete _mjesec; delete _godina;
_dan = _mjesec = _godina = nullptr;
}
};
struct Ispit{
char * _nazivPredmeta;
Datum _datumOdrzavanja;
char * _napomena;
int _ocjena;
void Unos(char * nazivPredmeta, Datum datumOdrzavanja){
int size = strlen(nazivPredmeta) + 1;
_nazivPredmeta = new char[size];
strcpy_s(_nazivPredmeta, size, nazivPredmeta);
_datumOdrzavanja.Unos(*datumOdrzavanja._dan,
*_datumOdrzavanja._mjesec, *datumOdrzavanja._godina);
_ocjena = 0;//ocjena se postavlja tek nakon sto student polozi ispit
_napomena = nullptr;
}
void PostaviOcjenu(int ocjena){ _ocjena = ocjena; }
void PostaviNapomenu(char * napomena){ _napomena = napomena; }
void Ispis(){
cout << _nazivPredmeta << "(" << _ocjena << ") Odrzan: ";
_datumOdrzavanja.Ispis(); }
void Dealociraj(){
delete[]_nazivPredmeta; _nazivPredmeta = nullptr;
delete[]_napomena; _napomena = nullptr;
_datumOdrzavanja.Dealociraj();
}
};
struct Student{
char * _ime;
char * _prezime;
char * _korisnickoIme;
char * _lozinka;
Ispit * _listaPrijavljenihIspita;
int _trenutnoPrijavljenih;
void Unos(char * ime, char * prezime, char * lozinka){
int size = strlen(ime) + 1;
_ime = new char[size];
strcpy_s(_ime, size, ime);
size = strlen(prezime) + 1;
_prezime = new char[size];
strcpy_s(_prezime, size, prezime);
_korisnickoIme = FormirajKorisnickoIme();
if (ProvjeriLozinku(lozinka)){
size = strlen(lozinka) + 1;
_lozinka = new char[size];
strcpy_s(_lozinka, size, lozinka);
}
else
_lozinka = nullptr;
_trenutnoPrijavljenih = 0;
_listaPrijavljenihIspita = nullptr;
}
void Dealociraj(){
delete[]_ime; _ime = nullptr;
delete[]_prezime; _prezime = nullptr;
delete[]_korisnickoIme; _korisnickoIme = nullptr;
delete[]_lozinka; _lozinka = nullptr;
delete[] _listaPrijavljenihIspita;
```

```
}
void Ispis(){
cout << _ime << " " << _prezime << " - " << _korisnickoIme << endl;
for (size_t i = 0; i < _trenutnoPrijavljenih; i++)
_listaPrijavljenihIspita[i].Ispis();
}
bool DodajPrijavu(Ispit * prijavaIspita){
/*dodaje podatke o novoj prijavi ispita. tokom skolovanja student identican predmet moze polagati najvise 6 puta, odnosno 3 puta ukoliko je na ispitu iz tog predmeta izvrsio neki od prestupa (npr. prepisivanje). prepisivanje i drugi prestupi se evidentiraju u okviru napomene na ispitu. u slucaju prijave ispita predmeta koji je polozen (skalu prolaznosti odredite samostalno) izvrsiti ponistavanje ranije ocjene i omoguciti novo polaganje. */
}
bool ProvjeriLozinku(char * lozinka){
/*Validna lozinka treba postovati sljedeca pravila:
a: pocinje sa najmanje jednim, a najviše sa tri broja
b: sadrzi najmanje 5, a najvise 14 znakova
c: predstavlja kombinaciju malih i velikih slova, brojeva i najmanje dva specijalna znaka (po vasem izboru, npr. '*', '_', '-')
d: ne smije sadrzavati ime ili prezime */
}
char * FormirajKorisnickoIme(){
/* funkcija je zaduzena za formiranje (generisanje) korisnickog imena koje treba biti u formatu: ime.prezime npr. indira.hamulic. korisnicko ime moze biti sastavljeno samo od malih slova, odnosno, nisu dozvoljena velika slova, brojevi ili specijalni znakovi */
}

};
Student * Pronadji(Student * s, int max, Datum OD, Datum DO, char * prestup){
/* funkcija vraca listu studenata koji su u odredjenom periodu (OD - DO) izvrsili definisani prestup (npr: prepisivao, izbacen, nije posjedovao validnu dokumentaciju) na nekom od prijavljenih ispita */
}
void main(){
//izbjegavajte koristenje vlastitog imena i prezimena.
//provjeriti validnost izvrsenja svih dostupnih funkcionalnosti
}
```

Posted by: Denis Mušić - Friday, January 22, 2016 11:00:02 AM

1. ZADATAK
Kakav ispis možemo očekivati kao rezultat izvršenja narednog programa? Navedite samo vrijednosti koje će program ispisati.

```
#include<iostream>
using namespace std;
int RekFunkcija(int * niz, int brojac){
int broj;
if (brojac == 1)
return niz[0];
broj = RekFunkcija(niz + 1, brojac - 1);
if (brojac < niz[broj])
return brojac++;
return niz[brojac++];
}
void main(){
char tekst[] = "Funkcija pa jos i rekurzivna";
int niz[] = { 9, -8, 1, 8, -7, 7 };
char * pok = tekst + RekFunkcija(niz, 6);
cout << pok << endl; //1. ispisuje _____________
cout << pok[RekFunkcija(niz, 3)] << endl;//2. ispisuje _____________
char * pok2 = &pok[RekFunkcija(niz, 3)];
cout << pok2 << endl; //3. ispisuje _____________
}
```

2. ZADATAK
Koristeći prikazane strukture izvršiti definiciju funkcija na način koji odgovara opisu (komentarima) samih funkcija. Također, dozvoljeno je dati komentar na bilo koju liniju code-a koju bi trebalo unaprijediti ili da će eventualno uzrokovati grešku prilikom kompajliranja.

```
#include<iostream>
using namespace std;
enum VrstaObaveze{ Seminarski, Parcijalni1, Parcijalni2, Integralni, Prakticni };
struct Datum{
int * _dan, *_mjesec, *_godina;
void Unos(int d, int m, int g){
_dan = new int(d);
_mjesec = new int(m);
_godina = new int(g);
}
void Ispis(){cout<< *_dan << "/"<< *_mjesec <<"/"<< *_godina << endl; }
void Dealociraj(){ delete _dan; delete _mjesec; delete _godina; }
};
struct ObavezeNaPredmetu{
VrstaObaveze _vrstaObaveze;
Datum * _datumIzvrsenja;
char * _napomena;
int _ocjena; // 5 - 10
void Unos(VrstaObaveze vrsta,Datum * datum,int ocjena,char * napomena){
```

```
_vrstaObaveze = vrsta;
_datumIzvrsenja = new Datum;
_datumIzvrsenja->Unos(*datum->_dan, *datum->_mjesec, *datum->_godina);
_ocjena = ocjena;
int size = strlen(napomena) + 1;
_napomena = new char[size];
strcpy(_napomena, napomena);
}
void Ispis(){
cout << _vrstaObaveze << " " << _ocjena << " " << _napomena;
_datumIzvrsenja->Ispis();
cout << endl;
}
void Dealociraj(){ delete _datumIzvrsenja; }
};
struct PolozeniPredmet{
Datum _datumPolaganja;
ObavezeNaPredmetu * _listaIzvrsenihObaveza[10];
int _trenutnoIzvrsenihObaveza;
int _konacnaOcjena; //formira se na osnovu ocjena izvrsenih obaveza

void Unos(Datum * d){
/*Na osnovu vrijednosti primljenog parametra izvrsiti inicijalizaciju odgovarajucih atributa*/
}

bool DodajIzvrsenuObavezu(ObavezeNaPredmetu o){
/*Na osnovu vrijednosti primljenog parametra osigurati dodavanje novoizvrsene obaveze na predmetu. Potrebno je onemoguciti dodavanje identicnih obaveza, a izmedju izvrsenja pojedinih obaveza mora proci najmanje 7 dana. Identicna vrsta obaveze se moze dodati samo u slucaju da se radi o Seminarskom ili je prethodno dodana obaveza (identicne vrste) imala ocjenu 5. Ukoliko je moguce, osigurati prosirenje niza*/
}

int FormirajKonacnuOcjenu(){
/*Konacna ocjene predstavlja prosjecnu ocjenu na predmetu, a za njeno formiranje student mora posjedovati polozen integralni ili dva parijcalna ispita. Ukoliko je ispit polozen putem parcijalnih ispita, student takodjer mora imati pozitivno (ocjenom vecom od 5) ocijenjena najmanje dva seminarska rada. U slucaju da bilo koji od navedenih uslova nije zadovoljen konacna ocjena treba biti postavljena na vrijednost 5. Konacna ocjena, takodjer, ne smije biti formirana u slucaju da u napomeni od dvije obaveze stoji rijec 'prepisivao' ili 'izbacen'.*/
}
};

int Pretraga(PolozeniPredmet * p, int max){
/*Funkcija ima zadatak da ispise informacije o svim polozenim predmetima kod kojih je student sve obaveze na predmetu izvrsio prije odredjenog datuma i ostvario konacnu ocjenu vecu od definisane (ocjenu i datum unosi korisnik). Nakon ispisa, funkcija vraca prosjecnu ocjenu polozenih predmeta koji su zadovoljili pomenute kriterije.*/
}

void main(){
cout<<"Izvinjavam se na kasnjenju post-a. Bio sam ubijedjen da sam postavio zadatak...."<<endl;
//izbjegavajte koristenje vlastitog imena i prezimena.
//provjeriti validnost izvrsenja svih dostupnih funkcionalnosti
}
```

Posted by: Deni Miličević - Thursday, September 10, 2015 9:24:09 PM

Profesore, možete li objaviti rekurziju sa julskog roka i zadatke sa danasnjeg ispita. Hvala unaprijed :)

Posted by: Denis Mušić - Wednesday, August 05, 2015 8:18:05 AM

```
#include <iostream>
using namespace std;

struct Datum{
int _dan, _mjesec, _godina;
void Unos(int d, int m, int g){_dan = d; _mjesec = m; _godina = g;}
void Ispis(){cout <<_dan << "/" << _mjesec << "/" <<_godina<<endl;}
};
struct Kandidat{
char * _imePrezime;
char _JMBG[14];
void unos(char * ip, char jmbg[]){
_imePrezime = new char[strlen(ip) + 1];
strcpy(_imePrezime, ip);
strcpy(_JMBG, jmbg);
}
void Dealociraj(){
delete[] _imePrezime; _imePrezime = nullptr;
}
void Info(){
cout << _imePrezime << " " << _JMBG << " " << endl;
}
};
```

```
struct Pitanje{
char * _tekstPitanja;
char * _odgovori[10];//maksimalno 10 odgovora
int _tacan;//lokacija tacnog odgovora - svako pitanje moze
imati samo jedan tacan odgovor
int _bodova; //broj bodova koje nosi pitanje

/*IMPLEMENTIRATI SLJEDEĆE FUNKCIJE:
1.Unos:: na osnovu vrijednosti primljenih parametara, inicijalizovati vrijednosti atributa strukture
2.AddOdgovor:: dodaje novi odgovor u listu ponudjenih odgovora. Voditi racuna da novi odgovor moze biti tacan
3.RemoveOdgovor:: na osnovu primljene lokacije u nizu uklanja podatke o ponudjenom odgovoru.Prilikom uklanjanja ocuvati redoslijed dodavanja odgovora.Uklanjanje tacnog odgovora zahtijeva odabir novog tacnog dogovora osim u slucaju da se uklanja posljednji ponudjeni
odgovor */
};
struct PrijemniIspit{
Datum * _datumOdrzavanja;
//kandidati prijavljeni za prijemni ispit
Kandidat * _prijavljeniKandidati[100];
//uspjeh ostvaren na prijemnom ispitu za svakog pojedinog
//kandidata
float * _uspjehKandidata[100];
//prag prolaznosti na prijemnom ispitu npr. 60 procenata
int _pragProlaznosti;

/*IMPLEMENTIRATI SLJEDEĆE FUNKCIJE:
1.Unos::na osnovu vrijednosti primljenih parametara, inicijalizovati vrijednosti atributa strukture
2.DodajKandidata::dodaje novog kandidata za polaganje prijemnog ispita. Onemoguciti dodavanje dva ista kandidata
3.PokreniIspit::na osnovu primljenih parametara (JMBG kandidata i niz pitanja) funkcija pokrece prijemni ispit i kandidatu omogucava unos odgovora na postavljena pitanja. Na kraju izvrsenja funkcije potrebno je informisati kandidata o tome da li je polozio prijemni ispit ili ne
*/
};
int main(){
/*
1. provjeriti validnost izvrsenja svih dostupnih funkcionalnosti.
2. izbjegavajte koristenje vlastitog imena i prezimena
*/
return 0;
}
```

Posted by: Vlado Bulić - Friday, July 24, 2015 11:37:29 AM

Profesore, možete li objaviti zadatke sa proslog roka, 6.7.2015?

Posted by: Edin Zukanović - Wednesday, July 01, 2015 8:49:25 PM

Nisam siguran da li je ok ali evo ga moje rješenje prvog zadatka. Komentari i sugestije dobrodošle :)

http://pastebin.com/1hLgFMuL

Posted by: Denis Mušić - Thursday, June 18, 2015 7:57:25 AM

```
#include<iostream>
using namespace std;

void Funkcija(int broj, char a, char b, char c){
if (broj == 0)
return;
Funkcija(broj - 1, a, c, b);
cout << a << " -> " << c << endl;
Funkcija(broj - 1, b, a, c);
}
void main(){
Funkcija(3, 'A', 'B', 'C');
}
```

Posted by: Denis Mušić - Thursday, June 18, 2015 7:56:56 AM

```
#include<iostream>

using namespace std;

enum Kompanija{ Pegasus, TurkishAirlines, AustrianAirlines, FlyEmirates };

int ID = 1;

struct Putnik{
```

```
        int _putnikID;
        char * _imePrezime;
        float _brojPredjenihMilja;
        void Unos(char * imePrezime){
               int size = strlen(imePrezime) + 1;
               _imePrezime = new char[size];
               strcpy_s(_imePrezime, size, imePrezime);
               _brojPredjenihMilja = 0;
               _putnikID = ID++;
        }
        void Info(){ cout << "[" << _putnikID<<"] "<< _imePrezime <<
                     " (" << _brojPredjenihMilja << " milja)" << endl; }

        void Dealociraj(){ delete[] _imePrezime; _imePrezime = nullptr; }

        void DodajPredjeneMilje(int brojMilja){ _brojPredjenihMilja += brojMilja; }
};

struct Rezervacija{
        Putnik * _putnik;
        int _oznakaSjedista;
        float _cijena;
        void Unos(Putnik putnik, int oznaka, float cijena){
               _putnik.Unos(putnik._imePrezime);
               _oznakaSjedista = oznaka;
               _cijena = cijena;
        }
        void Info(){
               _putnik.Info();
               cout << "Sjediste: " << _oznakaSjedista << " Cijena: "
                    << _cijena << endl;
        }
        void Dealociraj(){ _putnik.Dealociraj(); }
};
struct Let{
        Kompanija * _kompanija;
        char * _pocetak; //pocetna lokacija
        char * _destinacija;
        Rezervacija * _rezervacije;
        int _brojMjesta; //maksimalan broj mjesta na letu
        float _brojMilja; //odnosi se na duzinu leta – broj predjenih milja
```

```
        float _cijenaKarte;
        void Unos(Kompanija kompanija, char * pocetak, char * destinacija,
                  int brojMjesta, float brojMilja, float cijena){
                _kompanija = kompanija;
                int size = strlen(pocetak) + 1;
                _pocetak = new char[size];
                strcpy_s(_pocetak, size, pocetak);
                size = strlen(destinacija) + 1;
                _destinacija = new char[size];
                strcpy_s(_destinacija, size, destinacija);
                _brojMjesta = brojMjesta;
                _rezervacije = new Rezervacija[_brojMjesta];
                Putnik * temp;
                temp.Unos("<SLOBODNO MJESTO>");
                for (int i = 0; i < _brojMjesta; i++)
                        _rezervacije[i].Unos(temp, i, 0);
                _brojMilja = brojMilja;
                _cijenaKarte = cijena;
        }
        void Dealociraj(){
/*definisati funkciju vodeci racuna o oslobadjanju svih resursa koji su alocirani
za potrebe objekta tipa Let*/  }
        bool AddRezervaciju(Putnik * p){
/*na samom pocetku, funkcija treba ispisati listu svih SLOBODNIH sjedista na letu, te omoguciti odabir nekog do njih. prilikom formiranja cijene karte voditi racuna o broju predjenih milja; ako je putnik presao od  10000 do 50000 milja


        bool RemoveRezervacija(int oznakaSjedista){
/*funkcija uklanja rezervaciju na sjedistu cija je oznaka proslijedjena kao parametar funkcije, te vraca vrijednost. U slucaju da oznaceno mjesto nije rezervisano ili ne postoji na tom letu, funkcija vraca vrijednost false */}


};


float GetMiljeByKompanija(Let * letovi, int maxLetova, Kompanija k, Putnik p){
/*funkcija vraca broj milja koje je putnik p ostvario putujuci sa kompanijom k*/
}
void main(){
/*provjeriti validnost izvrsenja svih dostupnih funkcionalnosti. vrijednosti koje koristite prilikom testiranja mogu biti predefinisane tj. ne morate omogucavati korisniku da ih inicijalizuje.*/

}
```