# Zadaća 4.

***Ova zadaća nosi ukupno 6 poena. Zadaci 1, 3 i 5 nose po 1,3 poena, dok zadaci 2, 4 i 6 zadatak nose po 0,7 poena (zadaci 4 i 6 su mala prepravka zadataka 3 i 5 i smiju se kopirati). Svi zadaci se mogu uraditi na osnovu gradiva sa prvih jedanaest predavanja i pretpostavljenog predznanja iz predmeta "Osnove računarstva". Rok za predaju ove zadaće je utorak, 7. VI 2022. do kraja dana.***

NAPOMENA: U slučaju da smatrate da Vam u zadacima trebaju neke pomoćne funkcije za realizaciju neophodnih funkcionalnosti (a trebale bi Vam, ako ne želite programe sa više od 500 linija koda), možete ih slobodno dodati u privatni dio klase. Međutim, interfejs klase *ne smijete mijenjati*, osim ukoliko se postavke zadatka jasno ne vidi da treba praviti izmjene u interfejsu klase.

1. Definirajte i implementirajte klasu "`Kugla`" koja omogućava čuvanje podataka koji opisuju neku kuglu u prostoru. Kugla je opisana sa četiri podatka: $x$, $y$ i $z$ koordinatom centra i poluprečnikom $r$, koji mora biti nenegativan broj (dozvoljava se da bude $r = 0$; u tom slučaju, kugla se degenerira u tačku). Klasa treba da ima sljedeći interfejs:

```
explicit Kugla(double r = 0);
Kugla(double x, double y, double z, double r = 0);
explicit Kugla(const std::tuple<double, double, double> &centar, double r = 0);
double DajX() const;
double DajY() const;
double DajZ() const;
std::tuple<double, double, double> DajCentar() const;
double DajPoluprecnik() const;
double DajPovrsinu() const;
double DajZapreminu() const;
Kugla &PostaviX(double x);
Kugla &PostaviY(double y);
Kugla &PostaviZ(double z);
Kugla &PostaviCentar(double x, double y, double z);
Kugla &PostaviCentar(const std::tuple<double, double, double> &centar);
Kugla &PostaviPoluprecnik(double r);
void Ispisi() const;
void Transliraj(double delta_x, double delta_y, double delta_z);
friend bool DaLiSuIdenticne(const Kugla &k1, const Kugla &k2);
friend bool DaLiSuPodudarne(const Kugla &k1, const Kugla &k2);
friend bool DaLiSuKoncentricne(const Kugla &k1, const Kugla &k2);
friend bool DaLiSeDodirujuIzvana(const Kugla &k1, const Kugla &k2);
friend bool DaLiSeDodirujuIznutra(const Kugla &k1, const Kugla &k2);
friend bool DaLiSePreklapaju(const Kugla &k1, const Kugla &k2);
friend bool DaLiSeSijeku(const Kugla &k1, const Kugla &k2);
bool DaLiSadrzi(const Kugla &k) const;
friend double RastojanjeCentara(const Kugla &k1, const Kugla &k2);
```

   Prvi konstruktor kreira kuglu zadanog poluprečnika sa centrom u koordinatnom početku, dok naredna dva konstruktora kreiraju kuglu sa zadanom lokacijom centra i zadanim poluprečnikom. Pri tom se u drugom konstruktoru koordinate centra zadaju kao tri odvojena realna broja, a u trećem kao uređena trojka realnih brojeva. Dopušta se da poluprečnik ima vrijednost 0 (u tom slučaju se kugla degenerira u tačku), što je ujedno i podrazumijevana vrijednost ukoliko se ne zada vrijednost poluprečnika. U slučaju da se kao vrijednost poluprečnika zada negativan broj, treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Ilegalan poluprecnik". Metode "`DajX`", "`DajY`", "`DajZ`", "`DajCentar`", "`DajPoluprecnik`", "`DajPovrsinu`" i "`DajZapreminu`" redom vraćaju $x$, $y$, odnosno $z$ koordinatu kugle, zatim centar kugle (u vidu uređene trojke koordinata), te iznos poluprečnika, površine odnosno zapremine kugle. Metode "`PostaviX`", "`PostaviY`" i "`PostaviZ`" omogućavaju nezavisnu izmjenu individualnih koordinata centra već kreirane kugle. Sve ove metode vraćaju kao rezultat referencu na modificirani objekat, da bi se podržala mogućnost kaskadnog pozivanja (poput "`k.PostaviX(5).PostaviZ(3)`"). Pomoću metode "`PostaviCentar`" moguće je istovremeno postaviti sve koordinate centra. Metoda ima dvije verzije, prva prima individualne koordinate kao parametre, a druga uređenu trojku koordinata kao parametar. Metoda "`PostaviPoluprecnik`" omogućava naknadnu promjenu poluprečnika kugle, pri čemu se baca izuzetak ukoliko se pokuša postaviti negativan poluprečnik, slično kao u konstruktorima. Sve ove metode također vraćaju referencu na izmijenjeni objekat, da bi se podržala mogućnost

kaskadnog pozivanja. Metoda "`Ispisi`" ispisuje na ekran podatke o kugli u obliku "$\{(x,y,z),r\}$". Konačno, metoda "`Transliraj`" pomjera kuglu za iznos $\Delta x$ u smjeru $x$-ose, $\Delta y$ u smjeru $y$-ose i $\Delta z$ u smjeru $z$-ose, pri čemu se vrijednosti $\Delta x$, $\Delta y$ i $\Delta z$ navode kao parametri.

Predviđeno je i nekoliko prijateljskih funkcija koje ispituju međusobne odnose između kugli. Funkcija "`DaLiSuIdenticne`" vraća logičku vrijednost "tačno" ako i samo ako joj se kao parametri prenesu dvije identične kugle (kugle na istoj poziciji i s istim poluprečnikom), inače vraća logičku vrijednost "netačno". Funkcija "`DaLiSuPodudarne`" samo testira da li su kugle koje se zadaju kao parametri podudarne, odnosno da li imaju iste poluprečnike (pri tome im se pozicije mogu razlikovati), dok funkcija "`DaLiSuKoncentricne`" testira da li kugle imaju zajednički centar, dok im se poluprečnici mogu razlikovati. Dalje, slijede funkcije "`DaLiSeDodirujuIzvana`" odnosno "`DaLiSeDodirujuIznutra`" koje testiraju da li se kugle dodiruju. Pretpostavlja se da se kugle dodiruju ukoliko im rubovi (tj. sfere kojima su omeđene) imaju tačno jednu zajedničku tačku. Pri tome, dodir može biti izvana, ili iznutra (dodir je izvana ukoliko je tim kuglama ta zajednička tačka ujedno i jedina zajednička tačka). Navedene funkcije upravo testiraju ta dva tipa dodira. Funkcija "`DaLiSePreklapaju`" testira da li unutrašnjosti dvije kugle (unutrašnjost kugle čine sve njene tačke koje nisu na rubu) imaju zajedničkih tačaka, dok funkcija "`DaLiSeSijeku`" testira da li se rubovi dvije kugle sijeku, tj. imaju više od jedne zajedničke tačke (zapravo, kod kugli koje se sijeku, rubovi uvijek imaju beskonačno mnogo zajedničkih tačaka, od kojih sve leže na jednoj kružnici). Svake dvije kugle koje se sijeku također se i preklapaju, dok obrnuto ne mora vrijediti. Konačno, predviđena je i funkcija članica (metoda) "`DaLiSadrzi`" koja testira da li se kugla koji se zadaje kao njen parametar u potpunosti sadrži u kugli nad kojom je metoda pozvana (tj. da li je svaka njena tačka ujedno i tačka kugle nad kojim je metoda pozvana), te prijateljska funkcija "`RastojanjeCentara`" koja vraća rastojanje između centara dvije kugle koje joj se prenose kao parametri.

Napisanu klasu demonstrirajte u testnom programu koji traži da se tastature unese prirodan broj $n$, koji zatim treba kreirati prazan vektor od $n$ pametnih pokazivača na objekte tipa "`Kugla`". Nakon toga, sa tastature treba redom unositi podatke za $n$ kugli (podaci o svakoj kugli se unose posebno, prvo tri koordinate, a zatim i poluprečnik). Za svaku kuglu, nakon obavljenog unosa treba dinamički kreirati odgovarajuću kuglu inicijaliziranu u skladu s unesenim podacima i pametni pokazivač na tako kreiranu kuglu ubaciti u vektor. Ukoliko korisnik unese besmislene podatke (podatke koji nisu brojevi, ili negativan poluprečnik), treba ispisati poruku upozorenja i zatražiti novi unos podataka za istu kuglu. Nakon okončanja unosa, program treba translirati sve unesene kugle u skladu sa podacima koji se unose sa tastature. Za tu svrhu treba koristiti funkciju "`transform`" iz biblioteke "`algorithm`", pri čemu transformacionu funkciju koja se prosljeđuje funkciji "`transform`" treba izvesti kao lambda funkciju. Nakon obavljenih transformacija, treba sortirati sve kugle u rastući poredak po zapreminama (tj. kugla s manjom zapreminom dolazi prije kugle s većom zapreminom), te ispisati podatke o svim unesenim kuglama nakon obavljenih transformacija. Za sortiranje obavezno koristiti bibliotečku funkciju "`sort`" uz pogodno definiranu funkciju kriterija kao lambda funkciju, a za ispis treba koristiti funkciju "`foreach`" i prikladnu lambda funkciju. Potom treba ispisati podatke o kugli koja ima najveću površinu, za šta ćete iskoristiti funkciju "`max_element`" uz definiranje prikladne funkcije kriterija (ponovo kao lambda funkcije). Na kraju, program treba pronaći sve parove kugli koje se presjecaju i ispisati koje su to kugle (ili ispisati obavijest da takvih kugli nema). Ovo također trebate izvesti bez petlji nego kaskadnom primjenom funkcije "`foreach`", odnosno tako što će jedan poziv "`foreach`" funkcije koristiti lambda funkciju u čijem će se tijelu nalaziti poziv druge lambda funkcije. Ovo je okvirni opis šta testni program treba da radi, a precizan izgled dijaloga između korisnika i programa biće specificiran putem javnih autotestova.

NAPOMENA 1: U testnom programu se očigledno ne testiraju sve metode klase. To ne znači da one ostale metode koje nisu predviđene u testnom programu ne moraju raditi ispravno!

NAPOMENA 2: Može se činiti da implementacija ove klase zahtijeva dobro poznavanje analitičke geometrije u prostoru. Naprotiv, poznavanje formule za rastojanje dvije tačke i malo osnovne logike sasvim je dovoljno.

NAPOMENA 3: Mada ovaj zadatak ima dugačak opis, može se uraditi relativno brzo, jer sve tražene funkcije imaju vrlo kratke implementacije (jedan do dva reda koda).

NAPOMENA 4: U svim funkcijama u kojima se traži testiranje realnih brojeva na jednakost, uvjet $x = y$ zamijenite slabijim uvjetom $|x - y| < \varepsilon\ (|x|+|y|)$ gdje je $\varepsilon = 10^{-10}$.

2. Prepravite klasu "Kugla" iz prethodnog zadatka u klasu "NepreklapajucaKugla" koja ima praktično identičan interfejs kao i prethodna klasa, uz jedine razlike u interfejsu što je svuda "Kugla" zamijenjeno sa "NepreklapajucaKugla" i što su izbačene sve prijateljske funkcije osim funkcije "RastojanjeCentara", kao i funkcija članica "DaLiSadrzi". Objekti tipa "NepreklapajucaKugla" u suštini se razlikuju od objekata tipa "Kugla" po tome što se niti jedan objekat ovog tipa ne smije preklapati ni sa jednim drugim objektom istog tipa koji postoji u isto vrijeme (ovaj uvjet je zapravo najteži dio zadatka i nešto kasnije će biti objašnjeno kako ovo postići). Ukoliko se neki objekat ovog tipa koji upravo kreiramo preklapa sa nekim od objekata istog tipa koji u tom trenutku već postoje, treba baciti izuzetak tipa "logic_error" uz prateći tekst "Nedozvoljeno preklapanje". Recimo, ukoliko pokušamo kreirati tri objekta tipa "NepreklapajucaKugla" koristeći konstrukcije poput

```
NepreklapajucaKugla k1(2, 3, 1, 5);
NepreklapajucaKugla k2(10, 7, 8, 2);
NepreklapajucaKugla k3(4, 6, 3, 7);
```

treća definicija treba da baci izuzetak, jer se kugla sa centrom u tački (4, 6, 3) i poluprečnikom 7 preklapa sa ranije definiranom kuglom "k1" sa centrom u tački (2, 3, 1) i poluprečnikom 5. Kopiranje i međusobno dodjeljivanje objekata ovog tipa treba zabraniti (s obzirom da bi se kopija nekog objekta tipa "NepreklapajucaKugla" svakako preklapala sa njim samim).

Sve funkcije inspektori u klasi "NepreklapajucaKugla" ostaju posve identične kao u klasi "Kugla". Međutim, sve funkcije mutatori trebaju baciti isti izuzetak kao i konstruktor ukoliko promjena parametara kugle dovede do njenog preklapanja sa drugim već postojećim kuglama (pošto će se test na preklapanje izvoditi na više mjesta, najbolje je izvesti ga u nekoj pomoćnoj privatnoj funkciji koju ćete pozivati gdje god treba).

Napisanu klasu demonstrirajte u testnom programu koji traži da se tastature unese prirodan broj $n$, a zatim kreira prazan vektor čiji su elementi pametni pokazivači na kugle (tj. na objekte tipa "NepreklapajucaKugla"). Nakon toga, sa tastature treba redom unositi podatke za $n$ kugli (na isti način kao u prethodnom zadatku). Za svaku kuglu, nakon obavljenog unosa treba dinamički kreirati odgovarajuću kuglu inicijaliziranu u skladu sa unesenim podacima i pametni pokazivač na tako kreiranu kuglu ubaciti u vektor. Ukoliko konstrukcija ne uspije jer se nova kugla preklapa sa do tada unesenim kuglama, ili ukoliko su zadani besmisleni podaci, treba ispisati poruku upozorenja i zatražiti novi unos podataka za istu kuglu. Nakon okončanja unosa, program treba sortirati sve unesene kugle u rastući poredak po površini (tj. kugla s manjom površinom dolazi prije kugle sa većom površinom) i ispisati podatke o svim kuglama nakon obavljenog sortiranja. Za sortiranje obavezno koristite bibliotečku funkciju "sort" uz pogodno definiranu funkciju kriterija kao lambda funkciju.

Uputa: Najveći problem je kako detektirati da li se kugla koju kreiramo preklapa sa kuglama koje su već kreirane, odnosno kako konstruktor kugle koju kreiramo može znati za druge kugle. Jedno loše rješenje (koje nećete izvesti) je koristiti neki dijeljeni (statički) atribut koji bi bio recimo vektor pokazivača koji bi čuvao pokazivače na sve kreirane kugle). Međutim, postoji rješenje koje je mnogo bolje sa aspekta utroška resursa (koje trebate izvesti u ovoj zadaći). Svaki objekat tipa "NepreklapajucaKugla" će u sebi sadržavati jedan pokazivač na posljednju kuglu koji je kreirana prije njega (osim prve kreirane kugle koja će na tom mjestu imati nul-pokazivač), tako da će svi kreirani objekti tipa "NepreklapajucaKugla" faktički biti povezani u jednostruko povezanu listu (a sami objekti će biti čvorovi te liste). Pored toga, biće potreban i jedan statički atribut koji će sadržavati pokazivač na posljednju kreiranu kuglu (ili nul-pokazivač ukoliko niti jedna kugla nije kreirana). Ovaj atribut je potreban da bismo znali gdje počinje "lanac" povezanih kugli. Sad se test na preklapanje izvodi tako što se prođe kroz čitavu listu i testira preklapanje kugle koju razmatramo sa svim ostalim. Ukoliko testiranje prođe uspješno, novokreirana kugla se također "uvezuje" u listu. Interesantno je da će klasa "NepreklapajucaKugla" morati imati i destruktor, iako nigdje nema nikakve dinamičke alokacije memorije. Naime, kad objekat tog tipa prestane postojati, on mora sebe "isključiti" iz lanca. Obratite pažnju na specijalne slučajeve (tj. šta tačno treba ažurirati) kada se "isključuje" objekat koji se nalazi na jednom ili drugom kraju lanca!

3. Za potrebe rekonstrukcija Željeznica Bosne i Hercegovine, potrebno je napraviti program koji će vršiti automatsku najavu polazaka preko ozvučenja na željezničkoj stanici. Program treba najavljivati sve vozove koji odlase sa stanice u toku dana, kao i eventualna kašnjenja u polascima. Za tu svrhu, u programu je potrebno razviti dvije klase nazvane "Polazak" i "RedVoznje". Klasa

"Polazak" vodi evidenciju o jednom polasku, dok klasa "RedVoznje" vodi evidenciju o svim polascima u toku dana. Klasa "Polazak" ima sljedeći interfejs:

```
Polazak(std::string odrediste, int broj_voza, int broj_perona, bool brzi_voz,
  int sat_polaska, int minute_polaska, int trajanje_voznje);
void PostaviKasnjenje(int kasnjenje);
bool DaLiKasni() const;
int DajTrajanjeVoznje() const;
std::pair<int, int> DajcekivanoVrijemePolaska() const;
std::pair<int, int> DajOcekivanoVrijemeDolaska() const;
void Ispisi() const;
```

Objekti tipa "Polazak" čuvaju u sebi informaciju o nazivu odredišta, broju voza (cijeli broj s maksimalno 5 cifara), broju perona (cijeli broj u opsegu od 1 do 6), informaciju o tome da li je voz brzi voz ili ne, vremenu polaska (sati i minute), trajanju vožnje u minutama, kao i informaciju o eventualnom kašnjenju (također u minutama). Konstruktor inicijalizira sve atribute klase u skladu sa vrijednostima zadanim parametrima konstruktora, osim informacije o eventualnom kašnjenju, koja se automatski inicijalizira na 0. Konstruktor treba da baci izuzetak tipa "domain_error" uz odgovarajuće prateće tekstove (odredite ih po volji) ukoliko bilo koji od parametara ima besmislene vrijednosti. Metoda "PostaviKasnjenje" postavlja informaciju o eventualnom kašnjenju na vrijednost zadanu parametrom, dok metoda "DaLiKasni" omogućava da se sazna da li odgovarajuća vožnja kasni ili ne (metoda vraća logičku vrijednost "**true**" u slučaju kašnjenja, a logičku vrijednost "**false**" u suprotnom slučaju). Metoda "DajTrajanjeVoznje" daje kao rezultat trajanje vožnje u minutama, dok se pomoću metoda "DajOcekivanoVrijemePolaska" i "DajOcekivanoVrijemeDolaska" može saznati očekivano vrijeme polaska odnosno dolaska kada se uračuna iznos kašnjenja u odnosu na predviđeno vrijeme polaska/dolaska. Obje metode kao rezultat daju uređeni par, čija prva komponenta predstavlja sate, a druga komponenta minute. Konačno, metoda "Ispisi" treba da podrži ispis objekata tipa "Polazak" na ekran. U slučaju da se radi o polasku bez kašnjenja, ispis bi trebao da izgleda poput sljedećeg:

**Lokalni voz broj 3423, odredište Zenica, polazi s perona 2 u 15:40, a na odredište stiže u 17:10. Putnicima i voznom osoblju želimo ugodno putovanje.**

U stvarnom sistemu, ovaj tekst bi se trebao proslijediti sklopu za sintezu govora koji bi emitirao govornu informaciju preko ozvučenja, ali za potrebe ovog zadatka zadovoljićemo se prostim ispisom na ekran. Sati i minute se uvijek ispisuju kao dvocifreni brojevi, tako da se recimo 12 sati i 9 minuta ispisuje kao "12:09" a ne kao "12:9".

U slucaju da se radi o polasku koji kasni, ispis bi trebao da izgleda poput sljedećeg:

**Brzi voz broj 358, odrediste Čapljina, sa predviđenim vremenom polaska u 6:50, kasni oko 35 minuta, te će poći oko 7:25. Očekuje se da voz stigne na odredište oko 9:50. Izvinjavamo se putnicima zbog eventualnih neugodnosti.**

Klasa "RedVoznje" ima sljedeći interfejs:

```
explicit RedVoznje(int max_broj_polazaka);
RedVoznje(std::initializer_list<Polazak> lista_polazaka);
~RedVoznje();
RedVoznje(const RedVoznje &red_voznje);
RedVoznje(RedVoznje &&red_voznje);
RedVoznje &operator =(const RedVoznje &red_voznje);
RedVoznje &operator =(RedVoznje &&red_voznje);
void RegistrirajPolazak(std::string odrediste, int broj_voza, bool brzi_voz,
  int broj_perona, int sat_polaska, int minute_polaska, int trajanje_voznje);
void RegistrirajPolazak(Polazak *polazak);
int DajBrojPolazaka() const;
int DajBrojPolazakaKojiKasne() const;
int DajProsjecnoTrajanjeVoznji() const;
Polazak &DajPrviPolazak() const;
Polazak &DajPosljednjiPolazak() const;
void IsprazniRedVoznje();
void Ispisi(int sati, int minute) const;
```

Podaci o polascima se čuvaju u dinamički alociranim objektima tipa "Polazak", kojima se opet pristupa preko dinamički alociranog niza pokazivača na takve objekte. Alokacija tog niza

pokazivača vrši se iz konstruktora. Parametar konstruktora predstavlja maksimalan broj polazaka koji se mogu registrirati. Predviđen je i sekvencijski konstruktor, koji omogućava kreiranje objekata tipa "RedVoznje" iz liste inicijalizatora čiji su elementi tipa "Polazak". Destruktor oslobađa svu memoriju koja je zauzeta tokom života objekta, dok kopirajući konstruktor i kopirajući operator dodjele omogućavaju bezbjedno kopiranje i međusobno dodjeljivanje objekata tipa "RedVoznje" korištenjem strategije dubokog kopiranja. Također su predviđeni i pomjerajući konstruktor odnosno operator dodjele koji optimiziraju postupak kopiranja u slučajevima kada se kopiraju privremeni objekti. Metoda "RegistrirajPolazak" podržana je u dvije verzije. Prva verzija kreira novi polazak u skladu sa parametrima (koji su identični kao kod konstruktora klase "Polazak") i registrira ga u redu vožnje, dok druga verzija prosto kao parametar prihvata pokazivač na objekat tipa "Polazak" (za koji pretpostavljamo da je već na neki način kreiran) i registrira ga u redu vožnje. U oba slučaja, treba baciti izuzetak tipa "range_error" uz prateći tekst "Dostignut maksimalni broj polazaka" u slučaju da je dostignut maksimalan broj polazaka koji se mogu registrirati u redu vožnje. Metode "DajBrojPolazaka", "DajBrojPolazakaKojiKasne" i "DajProsjecnoTrajanjeVoznje" daju respektivno ukupan broj registriranih polazaka, broj polazaka koji kasne, te prosječno trajanje svih registriranih polazaka u minutama. Metodu "DajBrojPolazakaKojiKasne" trebalo bi realizirati uz pomoć funkcije "count_if" iz biblioteke "algorithm" uz definiranje prikladne funkcije kriterija kao lambda funkcije. Metode "DajPrviPolazak" i "DajPosljednjiPolazak" daju kao rezultat prvi i posljednji polazak (tj. objekat tipa "Polazak") u toku dana (uzimajući u obzir i eventualna kašnjenja). Obje metode vraćaju kao rezultat referencu da se izbjegne nepotrebno kopiranje objekata, i trebalo bi ih realizirati putem funkcija "min_element" i "max element" iz biblioteke "algorithm", također uz odgovarajuće funkcije kriterija realizirane kao lambda funkcije. U slučaju da nema registriranih polazaka, treba baciti izuzetak tipa "domain_error" uz prateći tekst "Nema registriranih polazaka". Metoda "IsprazniRedVoznje" uklanja sve registrirane polaske iz reda vožnje, tako da nakon poziva ove metode kolekcija treba biti u identičnom stanju kakva je bila neposredno nakon kreiranja. Konačno, metoda "Ispisi" ispisuje informacije o svim polascima, počev od zadanog vremena do kraja dana, sortiranu po očekivanim vremenima polijetanja (za sortiranje iskoristite funkciju "sort", uz pogodno definiranu funkciju kriterija izvedenu kao lambda funkciju). Ispis pojedinačnih polazaka vrši se prostim pozivom metode "Ispisi" nad objektima tipa "Polazak" pohranjenim u kolekciji.

Sve metode implementirajte izvan deklaracije klase, osim kratkih trivijalnih metoda koje trebate implementirati direktno unutar deklaracije klase. Obavezno napišite i testni program u kojem ćete testirati sve elemente napisanih klasa. Posebno se trebate uvjeriti kopirajući i pomjerajući konstruktor kopije i kopirajući i pomjerajući preklopljeni operator dodjele rade ispravno, kao i da ni u kom slučaju ne dolazi do curenja memorije.

4. Izmijenite program iz prethodnog zadatka tako da se u klasi "RedVoznje" za evidenciju pokazivača na dinamički alocirane objekte tipa "Polazak" umjesto dinamički alociranog niza pokazivača koristiti vektor čiji su elementi pametni pokazivači na objekte tipa "Polazak", čime uklanjamo i ograničenje na maksimalno mogući broj polazaka koji se mogu registrirati, te rješavamo probleme vezane za oslobađanje memorije. Samim tim, konstruktor klase "RedVoznje" više neće imati parametar, dok metode za registraciju polazaka više ne trebaju provjeravati da li je dostignut maksimalan broj polazaka, s obzirom da ograničenje na maksimalan broj polazaka više ne postoji. Također, druga verzija funkcije "RegistrirajPolazak" zahtijevaće kao parametar pametni a ne obični pokazivač na objekat tipa "Polazak". Razmislite sami šta treba da se desi sa destruktorom, kopirajućim i pomjerajućim konstruktorom i operatorima dodjele, te izvršite odgovarajuće izmjene. Obavezno testirajte da li sve radi ispravno nakon ovih izmjena.

5. Implementirajte jednostavan program koji omogućava vođenje administrativnih poslova u nekoj studentskoj biblioteci. Korisnike (klijente) biblioteke modelira struktura (ne klasa) "Student", koja sadrži atribute "broj_indeksa", "ime", "prezime" , "godina_studija", "adresa" i "telefon", koji redom sadrže podatke o imenu, prezimenu, broju indeksa, godini studija, adresi i broju telefona korisnika. Atributi "broj_indeksa" i "godina_studija" su cjelobrojni, dok su ostali atributi tipa "string". Knjige u biblioteci modelira klasa "Knjiga" čiji privatni atributi čuvaju informacije o naslovu knjige, imenu pisca, žanru i godini izdavanja, kao i informaciju o eventualnom zaduženju knjige. Naslov, ime pisca i žanr su tipa "string", godina izdavanja je cijeli broj, dok je informacija o zaduženju pokazivač na korisnika koji je zadužio knjigu, odnosno "**nullptr**" ukoliko knjiga nije zadužena. Interfejs klase sadrži konstruktor sa 4 parametra koji inicijalizira sve atribute klase na vrijednosti zadane parametrima (redoslijed parametara je isti kao i redoslijed gore navedenih

atributa), osim informacije o zaduženju koja se postavlja tako da signalizira da knjiga nije zadužena. Pored konstruktora, interfejs klase sadrži pristupne metode "`DajNaslov`", "`DajAutora`", "`DajZanr`", "`DajGodinuIzdavanja`", koje prosto vraćaju vrijednosti odgovarajućih atributa, te metode "`ZaduziKnjigu`", "`RazduziKnjigu`", "`DaLiJeZaduzena`", "`DajKodKogaJe`" i "`DajPokKodKogaJe`". Metoda "`ZaduziKnjigu`" vrši zaduživanje knjige, a parametar joj je referenca na studenta koji zadužuje knjigu. Metoda "`RazduziKnjigu`" nema parametara, a vrši razduživanje knjige. Slično njoj, metoda "`DaLiJeZaduzena`" također nema parametara, a vraća logičku informaciju da li je knjiga zadužena ili ne. Metoda "`DajKodKogaJe`" (isto bez parametara) daje kao rezultat referencu na studenta koji je zadužio knjigu, ili baca izuzetak tipa "`domain_error`" uz prateći tekst "Knjiga nije zaduzena" ukoliko knjiga nije zadužena. Slična je i metoda "`DajPokKodKogaJe`", samo što ona nikad ne baca izuzetak, nego vraća kao rezultat pokazivač na studenta koji je zadužio knjigu, ili "**`nullptr`**" ako knjiga nije zadužena.

Samu biblioteku modelira klasa "`Biblioteka`", koja objedinjuje u sebi podatke o svim korisnicima biblioteke, kao i svim članovima biblioteke. Radi brže pretrage, podaci se čuvaju u dvije mape, mapa korisnika i mapa knjiga, i one su jedini atributi ove klase. Svaki korisnik (student) i svaka knjiga imaju jedinstveni identifikacioni broj (broj indeksa odnosno evidencijski broj knjige), i oni su ključna polja ove dvije mape (tipa "**`int`**"). Ostatak podataka o korisnicima odnosno knjigama nalazi se u dinamički alocianim objektima tipa "`Student`" odnosno "`Knjiga`", tako da su pridružene vrijednosti u mapi knjiga odnosno mapi korisnika (obični) pokazivači na dinamički alocirane objekte koje sadrže podatke o odgovarajućem korisniku odnosno knjizi. Objekti tipa "`Biblioteka`" moraju se moći kreirati ne navodeći nikakve dopunske informacije, a također se mogu bezbjedno kopirati i međusobno dodjeljivati, koristeći strategiju dubokog kopiranja, pri čemu su predviđene optimizirane verzije u slučaju kada se kopiraju privremeni objekti. Također, primjerci ove klase treba da se brinu o oslobađanju svih resursa koje su zauzeli tokom njihovog života. Naravno, klasa "`Biblioteka`" treba podržavati odgovarajuće metode za manipulaciju sa podacima. Metoda "`RegistrirajNovogStudenta`" prima kao parametre podatke o novom korisniku biblioteke. Njeni Parametri su redom broj indeksa korisnika, ime, prezime, godina studija, adresa i broj telefona. Ova metoda kreira dinamički odgovarajući objekat tipa "`Student`" koji sadrži odgovarajuće podatke o korisniku i upisuje pokazivač na kreirani objekat u mapu pod zadanim brojem indeksa. U slučaju da već postoji student sa istim brojem indeksa, metoda baca izuzetak tipa "`logic_error`" uz prateći tekst "Vec postoji student s tim brojem indeksa". Metoda "`RegistrirajNovuKnjigu`" radi sličnu stvar, ali za knjige (tj. objekte tipa "`Knjiga`"). Parametri su evidencijski broj knjige, naslov, ime pisca, žanr i godina izdavanja, a prateći tekst uz izuzetak je "Knjiga s tim evidencijskim brojem vec postoji". Metoda "`NadjiKorisnika`" prima kao parametar broj indeksa korisnika biblioteke i vraća kao rezultat referencu na objekat koji sadrži prateće podatke o korisniku sa zadanim brojem indeksa, ili baca izuzetak tipa "`logic_error`" uz prateći tekst "Student nije nadjen" ako takvog korisnika nema. Sličnu stvar radi i metoda "`NadjiKnjigu`", ali za knjige (parametar je evidencijski broj, a prateći tekst uz izuzetak je "Knjiga nije nadjena"). Dalje, metoda "`IzlistajKorisnike`" ispisuje podatke o svim registriranim korisnicima, jedan za drugim, sa po jednim praznim redom između podataka o svaka dva korisnika, dok metoda "`IzlistajKnjige`" tu istu stvar radi za knjige. Podaci o pojedinim korisnicima odnosno knjigama imaju format poput sljedećeg:

**`Broj indeksa:`** *<broj_indeksa>*
**`Ime i prezime:`** *<ime>* *<prezime>*
**`Godina studija:`** *<godina_studija>*
**`Adresa:`** *<adresa>*
**`Broj telefona:`** *<broj_telefona>*

**`Evidencijski broj:`** *<evidencijski_broj>*
**`Naslov:`** *<naslov_knjige>*
**`Pisac:`** *<ime_pisca>*
**`Zanr:`** *<žanr>*
**`Godina izdavanja:`** *<godina_izdavanja>*

Pri tome, ukoliko je knjiga zadužena, treba još dodatno odmah ispod u novom redu ispisati i tekst "Zaduzena kod korisnika: " iza čega slijedi ime i prezime (sa jednim razmakom između) korisnika biblioteke koji je zadužio knjigu, te njegov broj indeksa unutar zagrada (npr. "Ibro Ibric (1234)"). Metoda "`ZaduziKnjigu`" prima kao parametre evidencijski broj knjige, kao i broj indeksa korisnika koji zadužuje knjigu. Ona vrši registraciju da je navedena knjiga zadužena kod navedenog korisnika. U slučaju da je evidencijski broj knjige ili broj indeksa korisnika neispravan, ili ukoliko

je knjiga već zadužena, metoda baca izuzetak tipa "`logic_error`" uz prateće tekstove "Knjiga nije nadjena", "Student nije nadjen", odnosno "Knjiga vec zaduzena" (prvo se testira knjiga pa korisnik, tako da ukoliko nisu ispravni niti evidencijski broj knjige niti broj indeksa korisnika, prijavljuje se prvi tekst). Metoda "`RazduziKnjigu`" prima kao parametar evidencijski broj knjige. Ona registrira da knjiga više nije zadužena. U slučaju da je evidencijski broj neispravan, ili ukoliko knjiga uopće nije bila zadužena, metoda baca izuzetak tipa "`logic_error`" uz prateće tekstove "Knjiga nije nadjena" odnosno "Knjiga nije zaduzena". Konačno, metoda "`PrikaziZaduzenja`" prima kao parametar broj indeksa korisnika, a vrši ispis podataka o svim knjigama koje je zadužio navedeni korisnik, na isti način kao u metodi "`IzlistajKnjige`" (bez prikazivanja ko je zadužio knjigu). U slučaju da korisnik nije zadužio niti jednu knjigu, metoda ispisuje tekst "Nema zaduzenja za tog studenta!", dok u slučaju da je broj indeksa neispravan, metoda baca izuzetak tipa "`logic_error`" uz prateći tekst "Student nije nadjen".

Sve funkcije koje logično trebaju biti inspektori obavezno deklarirajte kao takve. Napisane klase demonstrirajte u testnom programu u kojem se korisniku prikazuje meni koji mu nudi da odabere neku od mogućnosti koje su podržane u klasi "`Biblioteka`". Nakon izbora opcije, sa tastature treba unijeti eventualne podatke neophodne za izvršavanje te opcije, te prikazati rezultate njenog izvršenja. Ovo se sve izvodi u petlji dok korisnik programa ne izabere da želi završiti sa radom. Tačan izgled dijaloga između programa i korisnika osmislite po svojoj volji.

6. Neki studenti su pitali predmetnog nastavnika i druge članove nastavnog ansambla zašto ih maltretiramo sa ručnim upravljanjem alokacijom memorije i sličnim manuelnim tehnikama, kada postoje automatizirani postupci za slične stvari. Razlog je jednostavan: cilj je da se shvate i savladaju mehanizmi koji leže "ispod haube" tih automatiziranih postupaka. Ali, da se ne bi ljutili oni koji će reći "ali u praksi mi nećemo tako raditi", što je vjerovatno istina, u ovom zadatku ćete izvršiti prepravke prethodnog zadatka tako što ćete umjesto običnih pokazivača koristiti pametne pokazivače gdje god je to moguće (pri čemu na jednom mjestu to nije niti moguće, niti potrebno, a otkrijte sami gdje i zašto). Pri tome se objekti tipa "`Biblioteka`" i dalje trebaju kopirati kao duboke kopije, jer ovo je C++, a ne Java (imaćete prilike u Javi uživati u čarima plitkih kopija).