Zadaća 4.

Ova zadaća nosi ukupno 4 poena, pri čemu zadaci nose redom 1, 0.7, 1.2 i 1.1 poena. Svi zadaci se mogu uraditi na osnovu gradiva sa prvih 11 predavanja i pretpostavljenog predznanja iz predmeta "Osnove računarstva". Rok za predaju ove zadaće je nedjelja, 4. VI 2023. (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 vidi da treba praviti izmjene u interfejsu klase.

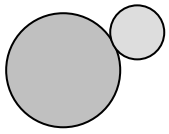
1. Definirajte i implementirajte klasu "`Krug`" koja omogućava čuvanje podataka koji opisuju neki krug u ravni. Krug je opisan pozicijom centra i dužinom poluprečnikom r, koji mora biti nenegativan broj (dozvoljava se da bude $r = 0$; u tom slučaju, krug se degenerira u tačku). Klasa treba da ima sljedeći interfejs:

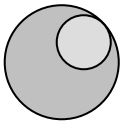
```
explicit Krug(double poluprecnik = 0);
explicit Krug(const std::pair<double, double> &centar, double poluprecnik = 0);
std::pair<double, double> DajCentar() const;
double DajPoluprecnik() const;
double DajObim() const;
double DajPovrsinu() const;
Krug &PostaviCentar(const std::pair<double, double> &centar);
Krug &PostaviPoluprecnik(double poluprecnik);
void Ispisi() const;
Krug &Transliraj(double delta_x, double delta_y);
Krug &Rotiraj(double alpha);
Krug &Rotiraj(const std::pair<double, double> &centar_rotacije, double alpha);
static double RastojanjeCentara(const Krug &k1, const Krug &k2);
static bool DaLiSuIdenticni(const Krug &k1, const Krug &k2);
static bool DaLiSuPodudarni(const Krug &k1, const Krug &k2);
static bool DaLiSuKoncentricni(const Krug &k1, const Krug &k2);
static bool DaLiSeDodirujuIzvani(const Krug &k1, const Krug &k2);
static bool DaLiSeDodirujuIznutri(const Krug &k1, const Krug &k2);
static bool DaLiSePreklapaju(const Krug &k1, const Krug &k2);
static bool DaLiSeSijeku(const Krug &k1, const Krug &k2);
bool DaLiSadrzi(const Krug &k) const;
friend Krug TransliraniKrug(const Krug &k, double delta_x, double delta_y);
friend Krug RotiraniKrug(const Krug &k, double alpha);
friend Krug RotiraniKrug(const Krug &k,
  const std::pair<double, double> &centar_rotacije, double alpha);
```

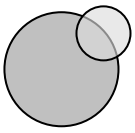
   Prvi konstruktor kreira krug zadanog poluprečnika sa centrom u koordinatnom početku, dok naredni konstruktor kreira krug sa zadanom lokacijom centra i zadanim poluprečnikom. Pri tome se lokacija centra zadaje kao uređeni par realnih brojeva koji predstavljaju Descartesove koordinate centra. Dopušta se da poluprečnik ima vrijednost 0 (u tom slučaju se krug 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 "Nedozvoljen poluprecnik". Metode "`DajCentar`", "`DajPoluprecnik`", "`DajPovrsinu`" i "`DajZapreminu`" redom vraćaju centar kugle (u vidu uređenog para koordinata), zatim iznos poluprečnika, te iznos obima odnosno površine kruga. Metode "`PostaviCentar`" i "`PostaviPoluprecnik`", omogućavaju izmjenu pozicije centra odnosno dužine poluprečnika već kreiranog kruga. One vraćaju kao rezultat referencu na modificirani objekat, da bi se podržala mogućnost kaskadnog pozivanja (poput "`k.PostaviCentar({2, 3}).PostaviPoluprecnik(10)`"). Pri tome, metoda "`PostaviPoluprecnik`" baca izuzetak ukoliko se pokuša postaviti negativan poluprečnik, slično kao u konstruktorima.. Metoda "`Ispisi`" ispisuje na ekran podatke o krugu u obliku "$\{(x,y),r\}$". Metoda "`Transliraj`" pomjera krug za iznos Δx u smjeru x-ose i za iznos Δy u smjeru y-ose, pri čemu se vrijednosti Δx i Δy navode kao parametri. Konačno, metoda "`Rotiraj`" dolazi u dvije verzije. Prva verzija rotira krug za ugao α koji se zadaje kao parametar (u smjeru suprotnom od kazaljke na satu) oko koordinatnog početka, dok druga verzija omogućava da se zada i tačka (u vidu uređenog para Descartesovih koordinata) oko koje se vrši rotacija. Za one koji ne znaju, a to su nažalost gotovo svi studenti koji će rješavati ovaj zadatak (što je, mora se istaći, velika sramota, ali se mora priznati i to da krivicu za tu sramotu ipak ne snose studenti),

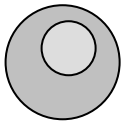
rotacijom tačke (x, y) oko tačke (x_c, y_c) za ugao α dobija se tačka (x', y'), gdje su x', y' dati kao $x' = x_c + (x - x_c)\cos\alpha - (y - y_c)\sin\alpha$ i $y' = y_c + (x - x_c)\sin\alpha + (y - y_c)\cos\alpha$). Opisane metode za translaciju i rotaciju također vraćaju kao rezultat referencu na modificirani objekat, da se podrži kaskadno pozivanje (poput "`k.Transliraj({5, 5}).Rotiraj(0.2)`"). Naravno, prilikom translacije i rotacije, mijenja se samo pozicija kruga, dok njegov poluprečnik ostaje isti.

Predviđeno je i nekoliko statičkih funkcija članica koje primaju objekte tipa "`Krug`" kao parametre. Na prvom mjestu, tu je funkcija "`RastojanjeCentara`" koja vraća rastojanje između centara dvije kugle koje joj se prenose kao parametri. Preostale statičke funkcije članice ispituju međusobne odnose između krugova. Tako, funkcija "`DaLiSuIdenticni`" vraća logičku vrijednost "tačno" ako i samo ako joj se kao parametri prenesu dva *identična* kruga (krugovi na *istoj poziciji* i s *istim poluprečnikom*), inače vraća logičku vrijednost "netačno". Funkcija "`DaLiSuPodudarni`" samo testira da li su krugovi koji se zadaju kao parametri *podudarni*, odnosno da li imaju *iste poluprečnike* (pri čemu im se *pozicije mogu razlikovati*), dok funkcija "`DaLiSuKoncentricni`" testira da li krugovi imaju *zajednički centar*, dok im se *poluprečnici mogu razlikovati*. Dalje, slijede funkcije "`DaLiSeDodirujuIzvana`" odnosno "`DaLiSeDodirujuIznutra`" koje testiraju da li se krugovi *dodiruju*. Pretpostavlja se da se krugovi dodiruju ukoliko im *rubovi* (tj. *kružnice* kojima su omeđeni) imaju *tačno jednu zajedničku tačku*. Pri tome, dodir može biti *izvana*, kao na Slici 1, ili *iznutra*, kao na Slici 2 (dodir je izvana ukoliko je tim krugovima *ta zajednička tačka ujedno i jedina zajednička tačka*). Navedene funkcije upravo testiraju ta dva tipa dodira. Funkcija "`DaLiSePreklapaju`" testira da li *unutrašnjosti dva kruga* (unutrašnjost kruga čine sve njegove tačke koje *nisu na rubu*) *imaju zajedničkih tačaka*, dok funkcija "`DaLiSeSijeku`" testira da li se *rubovi dva kruga sijeku*, tj. imaju *tačno dvije zajedničke tačke*. Svaka dva kruga koja se sijeku također se i preklapaju, dok obrnuto ne mora vrijediti. Recimo, na Slici 3. imamo dva kruga koja se sijeku (i preklapaju), dok se krugovi na Slici 4. preklapaju, ali se ne sijeku. Predviđena je i funkcija članica (metoda) "`DaLiSadrzi`" koja testira da li se krug koji se zadaje kao njen parametar u potpunosti sadrži u krugu nad kojim je metoda pozvana (tj. da li je svaka njegova tačka ujedno i tačka kruga nad kojim je metoda pozvana).


Slika 1


Slika 2


Slika 3


Slika 4

Na kraju, predviđene su i prijateljske funkcije "`TransliraniKrug`" i "`RotiraniKrug`" (ova posljednja u dvije verzije). Ove funkcije obavljaju sličan zadatak kao i funkcije članice "`Transliraj`" odnosno "`Rotiraj`", jedino što ne modificiraju krug nad kojim su pozvane, nego kao rezultat daju novi krug (tj. novi objekat tipa "`Krug`") koji se dobija kao rezultat translacije odnosno rotacije kruga koji im je poslan kao prvi parametar (on pri tome ostaje nepromijenjen).

Napisanu klasu demonstrirajte u testnom programu koji traži da sesa tastature unese prirodan broj n, koji zatim treba dinamički alocirati niz od n pokazivača na objekte tipa "`Krug`". Nakon toga, sa tastature treba redom unositi podatke za n krugova (podaci o svakom krugu se unose posebno, prvo dvije koordinate centra, a zatim i poluprečnik). Za svaki krug, nakon obavljenog unosa treba dinamički kreirati odgovarajući krug inicijaliziran u skladu s unesenim podacima i pokazivač na tako kreirani krug smjestiti u odgovarajući element dinamički alociranog niza. Ukoliko korisnik unese besmislene podatke (podatke koji nisu brojevi, ili negativan poluprečnik), treba ispisati poruku upozorenja i zatražiti novi unos podataka za isti krug. Nakon okončanja unosa, program treba prvo translirati sve unesene krugove, a zatim ih rotirati oko koordinatnog početka 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 krugove u rastući poredak po površinama (tj. krug s manjom površinom treba doći prije kruga s većom površinom), te ispisati podatke o svim unesenim krugovima nakon obavljenih transformacija. Za sortiranje obavezno koristite 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 krugu koji ima najveći obim, za šta ćete iskoristiti funkciju "`max_element`" uz definiranje prikladne funkcije kriterija (ponovo kao lambda funkcije). Na kraju, program treba pronaći sve parove krugova koji se presjecaju i ispisati koji su to krugovi (ili

ispisati obavijest da takvih krugova 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 ponovo nalaziti poziv "`foreach`" funkcije koja koristi drugu lambda funkciju. Naravno, treba predvidjeti i oslobađanje svih dinamički alociranih resursa prije završetka programa. 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. Naprotiv, poznavanje formule za rastojanje dvije tačke i malo osnovne logike sasvim je dovoljno za izvedbu svih traženih funkcionalnosti.

NAPOMENA 3: Mada ovaj zadatak ima dugačak opis, može se uraditi posve 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 $x \approx y$, koji ćete implementirati kao $|x-y| \leq \varepsilon(|x|+|y|)$ uz toleranciju $\varepsilon = 10^{-10}$. Također, uvjete $x \leq y$ odnosno $x \geq y$ zamijenite slabijim uvjetima $x < y$ ili $x \approx y$, odnosno $x > y$ ili $x \approx y$. Za potrebe testiranja uvjeta $x \approx y$ možete definirati pomoćnu privatnu statičku funkciju članicu.

2. Prepravite klasu "`Krug`" iz prethodnog zadatka u klasu "`NepreklapajuciKrug`" koja ima praktično identičan interfejs kao i prethodna klasa, uz jedine razlike u interfejsu što je svuda "`Krug`" zamijenjeno sa "`NepreklapajuciKrug`" i što su izbačene sve statičke funkcije članice osim funkcije "`RastojanjeCentara`", zatim funkcija članica "`DaLiSadrzi`", te prijateljske funkcije "`TransliraniKrug`" i "`RotiraniKrug`". Objekti tipa "`NepreklapajuciKrug`" u suštini se razlikuju od objekata tipa "`Krug`" po tome što se ni 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 "`NepreklapajuciKrug`" koristeći konstrukcije poput

```
NepreklapajuciKrug k1({2, 3}, 5);
NepreklapajuciKrug k2({10, 7}, 2);
NepreklapajuciKrug k3({4, 6}, 3);
```

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

Sve funkcije inspektori u klasi "`NepreklapajuciKrug`" ostaju posve identične kao u klasi "`Krug`". Međutim, sve funkcije mutatori trebaju baciti isti izuzetak kao i konstruktor ukoliko promjena parametara kruga dovede do njegoog preklapanja s drugim već postojećim krugovima (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 krugove (tj. na objekte tipa "`NepreklapajuciKrug`"). Nakon toga, sa tastature treba redom unositi podatke za n krugova (na isti način kao u prethodnom zadatku). Za svaki krug, nakon obavljenog unosa treba dinamički kreirati odgovarajući objekat tipa "`NepreklapajuciKrug`" inicijaliziranu u skladu sa unesenim podacima i pametni pokazivač na tako kreirani objekat ubaciti u vektor. Ukoliko konstrukcija ne uspije jer se novi krug preklapa sa do tada unesenim krugovima, ili ukoliko su zadani besmisleni podaci, treba ispisati poruku upozorenja i zatražiti novi unos podataka za isti krug. Nakon okončanja unosa, program treba sortirati sve unesene krugove u opadajući poredak po površini (tj. krug s većom površinom treba doći prije kruga s manjom površinom) i ispisati podatke o svim krugovima 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 krug koji kreiramo preklapa sa krugovima koji su već kreirani, odnosno *kako konstruktor kruga kojeg kreiramo može znati za druge krugove*. 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 krugove. Međutim, postoji rješenje koje je mnogo bolje sa aspekta utroška resursa (i koje *trebate izvesti u ovoj zadaći*). Svaki objekat tipa "`NepreklapajuciKrug`" će u sebi sadržavati jedan pokazivač na posljednji krug koji je kreiran prije njega (osim prvog kreiranog kruga koji će na tom mjestu imati nul-pokazivač), tako da će svi kreirani objekti tipa "`NepreklapajuciKrug`" 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 nijedan krug nije kreiran). 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, novokreirani krug se također "uvezuje" u listu. Interesantno je da će klasa "`NepreklapajuciKrug`" 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!

Napomena: Dozvoljeno je kopiranje izvjesnih dijelova koda iz prethodnog u ovaj zadatak.

3. Potrebno je napraviti program koji vrši najavu polazaka autobusa na displeju autobuske stanice. Program treba najavljivati sve polaske u toku dana, kao i eventualna kašnjenja u polascima. Za tu svrhu, u programu je potrebno razviti dvije klase nazvane "`Polazak`" i "`Polasci`". Klasa "`Polazak`" vodi evidenciju o jednom polasku, dok klasa "`Polasci`" vodi evidenciju o svim polascima u toku dana. Klasa "`Polazak`" ima sljedeći interfejs:

```
Polazak(std::string odrediste, std::string oznaka_voznje,  int broj_perona,
  int sat_polaska, int minute_polaska, int trajanje_voznje);
void PostaviKasnjenje(int kasnjenje);
bool DaLiKasni() const;
int DajTrajanje() const;
void OcekivanoVrijemePolaska(int &sati, int &minute) const;
void OcekivanoVrijemeDolaska(int &sati, int &minute) const;
void Ispisi() const;
```

Objekti tipa "`Polazak`" čuvaju u sebi informaciju o nazivu odredišta, oznaku vožnje (npr. "CTS 109"), broju perona (cijeli broj u opsegu od 1 do 15), vremenu polaska (sati i minute), trajanju vožnje u minutama, kao i informaciju o eventualnom kašnjenju (također u minutama). Konstruktor inicijalizira objekat 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 ma 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 "tačno" u slučaju kašnjenja, a logičku vrijednost "netačno" u suprotnom slučaju). Metoda "`DajTrajanje`" daje kao rezultat zadano trajanje vožnje, dok metode "`OcekivanoVrijemePolaska`" i "`OcekivanoVrijemeDolaska`" omogućavaju da se sazna očekivano vrijeme polaska odnosno dolaska kada se uračuna iznos kašnjenja u odnosu na predviđeno vrijeme polaska/dolaska. Obje metode treba da smjeste sate i minute očekivanog vremena polaska/dolaska u dva cjelobrojna parametra koji joj se prosljeđuju. 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:

CTS 109 Bihać 7:30 15:10 5

Ovi podaci predstavljaju redom oznaku vožnje, naziv odredišta, vrijeme polaska, očekivano vrijeme dolaska i broj perona. Predvidite širinu od 10 mjesta za ispis oznake vožnje, 30 mjesta za naizv odredišta, po 10 mjesta za vrijeme polaska i dolaska, odnosno 8 mjesta za ispis perona. U slucaju da se radi o polasku koji kasni, ispis bi trebao da izgleda poput sljedećeg:

***APM 314 Mostar 14:30* (*Planirano 14:10, Kasni 20 min*)**

Oznaku vožnje, naziv odredišta i vrijeme polaska formatirajte kao i u prethodnom slučaju, a dopunske informacije pišite u produžetku iza vremena polaska. 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".

Klasa "`Polasci`" ima sljedeći interfejs:

```
explicit Polasci(int max_broj_polazaka);
Polasci(std::initializer_list<Polazak> lista_polazaka);
~Polasci();
Polasci(const Polasci &polasci);
Polasci(Polasci &&polasci);
Polasci &operator =(const Polasci &polasci);
Polasci &operator =(Polasci &&polasci);
void RegistrirajPolazak(std::string odrediste, std::string oznaka_voznje,
  int broj_perona, int sat_polaska, int minute_polaska, int trajanje_voznje);
void RegistrirajPolazak(Polazak *polazak);
int DajBrojPolazaka() const;
int DajBrojPolazakaKojiKasne() const;
Polazak &DajPrviPolazak();
Polazak DajPrviPolazak() const;
Polazak &DajPosljednjiPolazak();
Polazak DajPosljednjiPolazak() const;
void Ispisi() const;
void IsprazniKolekciju();
```

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 "`Polasci`" 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 "`Polasci`" 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 kolekciji, dok druga verzija prosto kao parametar prihvata pokazivač na objekat tipa "`Polazak`" (za koji pretpostavljamo da je već na neki način kreiran) i registira ga u kolekciji. 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 kolekciji. Metode "`DajBrojPolazaka`" i "`DajBrojPolazakaKojiKasne`" daju ukupan broj registriranih polazaka, osnosno broj polazaka koji kasne. Metodu "`DajBrojPolazakaKojiKasne`" treba 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. odgovarajući objekat tipa "`Polazak`") u toku dana (uključujući i eventualna kašnjenja). Obje metode postoje u dvije verzije. Nekonstantne verzije vraćaju reference na odgovarajuće objekte tipa "`Polazak`", što omogućava i primjenu mutatorskih funkcija nad vraćenim polaskom (poput "`k.DajPrviPolazak().PostaviKasnjenje(20)`", dok konstantne verzije vraćaju odgovarajuće bezimene kopije, i služe da bi se ove metode mogle pozivati i nad konstantnim objektima tipa "`Polasci`", a da pri tome bude onemogućena nehotična izmjena sadržaja konstantnog objekta. Ove funkcije treba realizirati putem funkcija "`min_element`" i "`max_element`" iz biblioteke "`algorithm`", također uz odgovarajuće funkcije kriterija realizirane kao lambda funkcije. Metoda "`Ispisi`" ispisuje kompletan spisak svih polazaka, počev od zadanog vremena do kraja dana, sortiranu po očekivanim vremenima polazaka (za sortiranje iskoristite funkciju "`sort`", uz pogodno definiranu funkciju kriterija izvedenu kao lambda funkciju). U spisak treba dodati i prikladno zaglavlje, tako da bi ispis mogao izgledati poput sljedećeg:

Vožnja Odredište Polazak Dolazak Peron

CTS 109 Bihać 7:30 15:10 5
APM 314 Mostar 14:30 (Planirano 14:10, Kasni 20 min)
SMT 291 Čekrčići 15:35 16:20 12

Ispis pojedinačnih polazaka vrši se prostim pozivom metode "`Ispisi`" nad objektima tipa "`Polazak`" pohranjenim u kolekciji. Konačno, metoda "`IsprazniKolekciju`" uklanja sve registrirane polaske, tako da nakon poziva ove metode kolekcija treba da bude u identičnom stanju kakva je bila neposredno nakon kreiranja. Konačno,

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. Tri studenta (dva mladića i djevojka) ulaze u Bosansku piramidu sunca u Visokom, tražeći dokaze koji će potvrditi ili pobiti teoriju o Bosni kao kolijevci civilizacije. Tako počinje njihovo putovanje kroz vrijeme i istoriju Bosne i Hercegovine. Naime, ono što je izgledalo kao obična rupa u padini brda pretvara se u vremenski tunel, tunel iz kojeg izlaz vodi uvijek u neki drugi dio bosanskohercegovačke istorije. Susreću Ivana de Kazamarisa, Aliju Sirotanovića, Gavrila Principa, Mula Mustafu Bašeskiju, Aleksu Šantića i Svetozara Ćorovića (i Eminu?), Valtera Perića, Branislava Nušića, borce ispod sarajevske piste…

   Gore napisani odlomak nije plod bolesne mašte predmetnog nastavnika, nego je sažetak romana "Vremenska petlja" autora Nenada Veličkovića (i pozorišne predstave sličnog naziva "Vremenski tunel"). Ipak, u ovom zadatku, moraćete također malo otploviti u prošlost, sa ili bez pomoći Bosanske piramide sunca. Nećete morati otploviti toliko daleko u prošlost kao naši junaci, nego svega možda nekih 15-ak godina unazad. Naime, ne tako davno, filmovi nisu bili tako dostupni putem interneta kao danas. Umjesto toga, gotovo svako naselje imalo je ponekad i nekoliko videoteka, u kojima su se filmovi mogli iznajmljivati za gledanje, slično kao knjige u biblioteci. Pri tome su filmovi mogli biti pohranjeni na analognom mediju (video trakama od magnetnog materijala pohranjenih u kasetama, među kojima su najpoznatije bile tzv. VHS trake) ili, u novije vrijeme, i na digitalnom mediju (optičkim DVD diskovima). DVD diskovi (i uređaji za reprodukciju istih) mogu se doduše (mada rijetko) susresti i danas, dok se VHS trake (i uređaji za reprodukciju istih) danas mogu naći samo u muzejima i na pokojoj izložbi starina.

   Uglavnom, spletom magičnih okolnosti, zadesili ste se u bliskoj prošlosti i dobili ste zadatak da implementirate jednostavni program koji olakšava vođenje administrativnih poslova u nekoj videoteci (prema proročanstvu faraona Ramiza III, uspješno rješavanje ovog zadatka Vam je osnovni preduvjet da se vratite u sadašnjost). Program se zasniva na tri klase "`Korisnik`", "`Film`" i "`Videoteka`". Primjerci ovih klasa modeliraju respektivno korisnike videoteke, filmove u videoteci, te samu videoteku.

   Klasa "`Korisnik`" sadrži privatne atribute koji čuvaju informacije o članskom broju korisnika, njegovom imenu i prezimenu (oboje u istom atributu), adresi, te broju telefona (svi ovi atributi su tipa "`string`", osim članskog broja koji je cijeli broj). Interfejs klase sadrži konstruktor sa četiri parametara koji inicijalizira sve atribute na vrijednosti zadane parametrima (redoslijed parametara je isti kao i redoslijed gore navedenih atributa), zatim odgovarajuće trivijalne pristupne metode "`DajClanskiBroj`", "`DajImeIPrezime`", "`DajAdresu`" i "`DajTelefon`" koje prosto vraćaju vrijednosti odgovarajućih atributa, te metodu "`Ispisi`" koja ispisuje podatke o korisniku na ekran. Ispis treba da izgleda ovako:

   Clanski broj: ***članski_broj***
   Ime i prezime: ***ime_i_prezime***
   Adresa: ***adresa***
   Telefon: ***broj_telefona***

   Klasa "`Film`" sadrži privatne atribute koji čuvaju informacije o evidencijskom broju video trake ili CD-a na kojem je film, zatim da li je film na video traci ili DVD-u, nazivu filma, žanru i godini produkcije, kao i informaciju o eventualnom zaduženju filma. Evidencijski broj i godina izdavanja su cijeli brojevi, informacija da li je film na video traci ili DVD-u je logičkog tipa, informacija o zaduženju čuva se kao pokazivač (obični) na korisnika koji je zadužio film odnosno nul-pokazivač ukoliko film nije zadužen, dok su ostali atributi stringovnog tipa. Interfejs klase sadrži konstruktor s 5 parametara 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 film nije zadužen. Pored konstruktora, interfejs klase sadrži trivijalne pristupne metode "`DajEvidencijskiBroj`", "`DajNaziv`", "`DajZanr`", "`DajGodinuProdukcije`" i koje vraćaju vrijednosti odgovarajućih atributa, metodu "`DaLiJeDVD`" koja vraća informaciju da li je film na DVD-u ili ne, te metode "`ZaduziFilm`", "`RazduziFilm`", "`DaLiJeZaduzen`", "`DajKodKogaJe`", "`DajPokKogaJe`" i "`Ispisi`". Metoda "`ZaduziFilm`" vrši zaduživanje filma, a parametar joj je referenca na korisnika koji zadužuje film. Metoda "`RazduziFilm`" nema parametara, a vrši razduživanje filma. Metoda "`DaLiJeZaduzen`" također nema parametara i prosto vraća informaciju da li je film zadužen ili ne. Metoda "`DajKodKogaJe`" (isto bez parametara) daje kao rezultat referencu na korisnika koji je zadužio film, ili baca izuzetak tipa "`domain_error`" uz prateći tekst "Film nije zaduzen" ukoliko film nije zadužen. Slična je i metoda "`DajPokKodKogaJe`", samo što ona nikad ne baca izuzetak, nego vraća kao rezultat pokazivač na korisnika koji je zadužio film, odnosno nul-pokazivač ako film nije zadužen. Konačno, metoda "`Ispisi`" vrši ispis podataka o filmu na ekran. Ispis treba da izgleda ovako:

Evidencijski broj: ***evidencijski_broj***
Medij: Video traka (*ili* DVD *ako je film na DVD-u*)
Naziv filma: ***naziv_filma***
Zanr: ***žanr***
Godina produkcije: ***godina_produkcije***

Klasa "`Videoteka`" objedinjuje u sebi podatke o svim korisnicima videoteke, kao i o svim filmovima. Podaci o svakom korisniku odnosno svakom filmu čuvaju se u dinamički alociranim objektima tipa "`Korisnik`" odnosno "`Film`". Radi brže i jednostavnije pretrage, klasa "`Videoteka`" kao atribute sadrži dvije mape, nazvane mapa korisnika i mapa filmova, i one su jedini atributi ove klase. Svaki korisnik videoteke i svaki film imaju jedinstveni identifikacioni broj (članski broj odnosno evidencijski broj filma), i oni su ključna polja ove dvije mape (tipa "`int`"). S druge strane, pridružene vrijednosti u mapi korisnika odnosno mapi filmova su pametni pokazivači (tipa "`shared_ptr`") na dinamički alocirane objekte koje sadrže podatke o odgovarajućem korisniku odnosno filmu. Objekti tipa "`Videoteka`" 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. Duboko kopiranje treba podržati jer je ovo C++, a ne Java (imaćete prilike u Javi uživati u čarima plitkih kopija). Destruktor neće biti potreban, jer će se pametni pokazivači pobrinuti za ispravno brisanje svih korisnika i filmova koji su evidentirani u videoteci (tj. koji su u vlasništvu objekta tipa "`Videoteka`").

Pored ovih elemenata, klasa "`Videoteka`" sadrži i nekoliko metoda, koje služe za manipuliranje s podacima o kojima videoteka vodi računa. Tako, metoda "`RegistrirajNovogKorisnika`" prima kao parametre podatke o korisniku (ovi parametri su isti kao parametri konstruktora klase "`Korisnik`"), nakon čega kreira odgovarajući objekat tipa "`Korisnik`" i upisuje ga u evidenciju (tačnije, pametni pokazivač koji pokazuje na kreirani objekat upisuje se u odgovarajuću mapu pod zadanim članskim brojem). U slučaju da već postoji korisnik s istim članskim brojem, metoda baca izuzetak tipa "`logic_error`" uz prateći tekst "Vec postoji korisnik s tim clanskim brojem". Metoda "`RegistrirajNoviFilm`" radi analognu stvar, ali za filmove (tj. objekte tipa "`Film`"). Parametri su isti kao za konstruktor klase "`Film`", a prateći tekst uz izuzetak je "Film s tim evidencijskim brojem vec postoji". Metoda "`NadjiKorisnkia`" prima kao parametar članski broj korisnika i vraća kao rezultat referencu na objekat koji sadrži prateće podatke o korisnuku sa zadanim članskim brojem, ili baca izuzetak tipa "`logic_error`" uz prateći tekst "Korisnik nije nadjen" ako takvog korisnika nema. Metoda "`NadjiFilm`" radi analognu stvar, ali za filmove (parametar je evidencijski broj, a prateći tekst uz izuzetak je "Film nije nadjen"). 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 "`IzlistajFilmove`" tu istu stvar radi za filmove. Podaci o pojedinim korisnicima odnosno filmovima ispisuju se pozivom metode "`Ispisi`" nad odgovarajućim objektima tipa "`Korisnik`" odnosno "`Film`". Pri tome, ukoliko je film zadužen, iza standardnih podataka koji se ispisuju za film treba u novom redu ispisati i tekst "Zadužen kod korisnika: " iza čega slijedi ime i prezime korisnika koji je zadužio film, te njegov članski broj unutar zagrada (npr. "Meho Mehic (1123)"). Metoda "`ZaduziFilm`" prima kao parametre evidencijski broj filma, kao i članski broj korisnika koji zadužuje film. Ona vrši registraciju da je navedeni film zadužen kod navedenog korisnika. U slučaju da je evidencijski broj filma ili članski broj korisnika neispravan,

ili ukoliko je film već zadužen, metoda baca izuzetak tipa "`logic_error`" uz prateće tekstove "Film nije nadjen", "Korisnik nije nadjen", odnosno "Film vec zaduzen" (prvo se testira film pa korisnik, tako da ukoliko nisu ispravni ni evidencijski broj filma ni članski broj korisnika, prijavljuje se prvi tekst). Metoda "`RazduziFilm`" prima kao parametar evidencijski broj filma. Ona registrira da film više nije zadužen. U slučaju da je evidencijski broj neispravan, ili ukoliko film uopće nije bio zadužen, metoda baca izuzetak tipa "`logic_error`" uz prateće tekstove "Film nije nadjen" odnosno "Film nije zaduzen". Konačno, metoda "`PrikaziZaduzenja`" prima kao parametar članski broj korisnika, a vrši ispis podataka o svim filmovima koje je zadužio navedeni korisnik, na isti način kao u metodi "`IzlistajFilmove`", ali bez prikazivanja ko je zadužio film. U slučaju da korisnik nije zadužio ni jedan film, metoda ispisuje tekst "Korisnik nema zaduzenja!", dok u slučaju da je članski broj neispravan, metoda baca izuzetak tipa "`logic_error`" uz prateći tekst "Korisnik 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 "`Videoteka`". 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.