Zadaci za samostalno vježbanje 11.

Svi zadaci dati ovdje su takvi da se mogu uraditi korištenjem isključivo gradiva prvih jedanaest predavanja i ranije stečenog predznanja na predmetu "Osnove računarstva". Zvjezdicom (*) su označeni zadaci koji su se pojavljivali na dosadašnjim ispitima (eventualno uz minorne modifikacije, koje suštinski ništa ne mijenjaju). Tarabom (#) su označeni zadaci koji su u prethodnim generacijama bili zadaci za zadaću.

1. Prikažite *tačan izgled ekrana* na kraju izvršavanja ovog C++ programa, uz kratko obrazloženje zbog čega su rezultati onakvi kakvi jesu. Bitan je *svaki razmak* i *svaki prelazak u novi red*.

```
#include <iostream>

class A {
  int x, y;
public:
  A() { std::cout << "A\n"; x = 1; y = 2; }
  A(int x, int y) { std::cout << "B\n"; A::x = x; A::y = y; }
  void w(int z) { x *= z; y *= z; }
  int u() { return x + 10; }
  int v() { return y - 5; }
  void s() { std::cout << x + y << " " << x - y << std::endl; }
};

class B {
  int x;
  A y;
public:
  B(int y) { std::cout << "C\n"; x = y; }
  B(int x, int y, int z) : y(x, z) { std::cout << "D\n"; B::x = y; }
  int u() { return x; }
  A v() { return y; }
  void s() { std::cout << x << std::endl; y.s(); }
};

int main() {
  A a; A b(3, 2);
  std::cout << a.u() << " " << a.v() << std::endl;
  std::cout << b.u() << " " << b.v() << std::endl;
  a.s(); b.s();
  A c = a;
  c.w(2); a.s(); c.s();
  B d(2); B e(1, 5, 3);
  std::cout << d.u() << " " << d.v().u() << " " << d.v().v() << std::endl;
  e.s();
  A f[4]; A g[3] = {{3, 2}, {1, 1}, {2, 2}};
  f[2].s(); g[1].s();
  A *p, *q;
  p = new A(10, 10); q = p;
  q->w(3); p->s();
  return 0;
}
```

2*. Za potrebe neke meteorološke stanice neophodno je vršiti čestu registraciju spoljašnje temperature. Za tu svrhu meteorološka stanica koristi računarski program u kojem je definirana i implementirana klasa nazvana "`Temperature`". Ova klasa omogućava čuvanje podataka o temperaturi za izvjesni vremenski period u dinamički alociranom nizu realnih brojeva kojem se pristupa preko nekog od internih atributa klase. Poznato je da klasa sadrži sljedeće elemente:

   a) Konstruktor sa jednim parametrom, koji vrši dinamičku alokaciju memorije, pri čemu parametar predstavlja maksimalan broj temperatura koje se mogu registrirati (pri tome je potrebno spriječiti da se taj konstruktor koristi za automatsku konverziju cjelobrojnih podataka u objekte tipa "`Temperature`").
   b) Destruktor, koji oslobađa memoriju koju su objekti tipa "`Temperature`" zauzeli tokom svog života.
   c) Konstruktor kopije koji će osigurati da se objekti tipa "`Temperature`" mogu bezbjedno prenositi kao parametri po vrijednosti, koristiti za inicijalizaciju drugih objekata istog tipa i vraćati kao rezultati iz funkcija, kao i preklopljeni operator dodjele, koji garantira sigurno dodjeljivanje jednog objekta tipa "`Temperature`" drugom objektu istog tipa.

d) Metodu koja vrši registraciju nove temperature, pri čemu se temperatura koja se registrira prenosi kao parametar metode (u slučaju da se dostigne maksimalan broj temperatura koje se mogu registrirati, treba baciti izuzetak).
e) Metodu koja briše sve unesene temperature.
f) Metode koje vraćaju kao rezultat prosječnu, minimalnu i maksimalnu temperaturu koje su registrirane u toku perioda posmatranja.
g) Metodu koja ispisuje sve unesene (registrirane) temperature sortirane u opadajućem poretku (tj. najveća temperatura se ispisuje prva), pri čemu se svaka temperatura ispisuje u posebnom redu.

Implementirajte klasu sa navedenim svojstvima. Sve neophodne atribute obavezno izvedite kao privatne članove klase, a sve metode implementirajte izvan klase, osim metoda čija implementacija zahtijeva jednu ili dvije naredbe. Obavezno napišite i mali testni program u kojem će se testirati sve navedene metode.

3. Izmijenite klasu "`Temperature`" razvijenu u prethodnom zadatku, tako što će se podaci o temperaturi umjesto u dinamički alociranom nizu čuvati u vektoru realnih brojeva (koji će biti privatni atribut klase). Ostatak interfejsa klase i njeno funkcioniranje treba da ostanu neizmijenjeni. Napomena: Ovim nestaje potreba za pisanjem vlastitog destruktora, konstruktora kopije i preklopljenog operatora dodjele, s obzirom da će podrazumijevani (automatski generirani) destruktor, konstruktor kopije i preklopljeni operator dodjele biti sasvim zadovoljavajući (s obzirom da se zahvaljujući odgovarajućim elementima implementiranim unutar klase "`vector`" objekti tipa "`vector`" ispravno uništavaju, kopiraju i dodjeljuju bez potrebe da programer o tome vodi računa).

4. Izmijenite klasu "`Temperature`" razvijenu u prethodnom zadatku, tako što ćete ukloniti ograničenje da se mora zadavati maksimalan broj temperatura koje se mogu registrirati (drugim riječima, maksimalan broj temperatura koje se mogu registrirati biće ograničen samo količinom raspoložive memorije). Ovim konstruktor klase više ne treba imati parametar (zahvaljujući fleksibilnosti vektora, moći će se dodati onoliko temperatura koliko želimo, bez potrebe da unaprijed specificiramo njihov maksimalan broj). Jasno je i da metoda za dodavanje nove temperature više ne treba provjeravati da li je dostignut maksimalan broj temperatura, s obzirom da ograničenje na maksimalan broj temperatura više ne postoji.

5*. Izmijenite klasu iz Zadatka 1. uz pretpostavku da dinamički alocirani niz ne sadrži samo po jednu temperaturu, nego po tri temperature, koje su dobijene na osnovu tri mjerenja temperature koje se redovno vrše u toku jednog dana. Elementi niza su po tipu strukture koje se sastoje od tri realna atributa, koji respektivno čuvaju jutarnju, podnevnu i večernju temperaturu za jedan dan. Metodu za unos nove temperature treba prepraviti da prima tri parametra koji respektivno predstavljaju jutarnju, podnevnu i večernju temperaturu za jedan dan. Metodu za ispis unesenih (registriranih) temperatura treba prepraviti, da se u svakom redu nalaze redom jutarnja, podnevna i večernja temperatura za svaki od dana, razdvojene zarezima (temperature za pojedine dane se ispisuju svaka u posebnom redu). Ispis treba da bude sortiran po prosječnoj dnevnoj temperaturi, tj. podaci za dan u kojem je bila veća dnevna temperatura treba da se ispišu prije podataka za dan sa manjom dnevnom temperaturom.

6. Načinite iste izmjene kao u Zadatku 4, ali sa klasom iz Zadatka 2.

7. Načinite iste izmjene kao u Zadatku 4, ali sa klasom iz Zadatka 3.

8#. Neka hidrometeorološka stanica za praćenje dnevnog vodostaja neke rijeke u toku određenog vremenskog perioda koristi računarski program u kojem je definirana i implementirana klasa nazvana "`Vodostaji`". Ova klasa omogućava čuvanje podataka o vodostaju posmatrane rijeke za u nekom od internih atributa klase koji predstavlja dinamički alocirani niz cijelih brojeva (vodostaj se izražava u centimetrima, zaokruženo na cijeli broj). Pored ovog atributa, poznato je da klasa sadrži sljedeće elemente:

a) Konstruktor sa jednim parametrom, koji vrši dinamičku alokaciju memorije i inicijalizaciju svih ostalih atributa, pri čemu parametar predstavlja maksimalan broj podataka o vodostaju koji se mogu registrirati (pri tome je potrebno spriječiti da se ovaj konstruktor koristi za automatsku konverziju cjelobrojnih podataka u objekte tipa "`Vodostaji`").
b) Destruktor koji oslobađa memoriju koju je alocirao konstruktor.

c) Metodu koja vrši registraciju novog podatka o vodostaju, pri čemu se podatak koja se registrira prenosi kao parametar metode (u slučaju da se dostigne maksimalan broj podataka koje se mogu registrirati, treba baciti izuzetak).
d) Metodu koja vraća kao rezultat prosječni vodostaj tokom posmatranog perioda.
e) Metodu koja briše sve unesene podatke.
f) Metode koje vraćaju broj dana u kojima je registriran vodostaj veći od vrijednosti koja je zadata kao parametar, kao i broj dana u kojima je registriran vodostaj manji od vrijednosti koja je zadata kao parametar (podrazumijeva se da svaki registrirani vodostaj odgovara jednom danu, tj. da se vodostaji registriraju jedanput dnevno).
g) Metodu koja ispisuje sve registrirane podatke sortirane u rastućem poretku (tj. najmanji vodostaj se ispisuje prvi), pri čemu se svaki podatak ispisuje u posebnom redu.

Kopiranje i međusobno dodjeljivanje objekata tipa "`Vodostaji`" treba zabraniti. Implementirajte klasu sa navedenim svojstvima. Sve neophodne atribute treba obavezno izvesti kao privatne članove klase, a sve metode implementirati izvan klase, osim metoda čija implementacija zahtijeva jednu ili dvije naredbe. Obavezno napišite i mali testni program u kojem će se testirati sve navedene metode.

9#. Za potrebe nekog fakulteta neophodno je vršiti evidenciju o polaznicima (studentima) i njihovom uspjehu. Za tu svrhu fakultet koristi računarski program u kojem je definirana i implementirana klasa nazvana "`Student`", kao i klasa nazvana "`Fakultet`". Klasa "`Student`" sadrži sljedeće elemente:

a) Privatne atribute koje čuvaju podatke o imenu i prezimenu studenta (po maksimalno 20 znakova), broju indeksa (cijeli broj), broju položenih ispita (cijeli broj) i prosječnoj ocjeni (realan broj), i nikave druge atribute.
b) Konstruktor čiji su parametri ime i prezime studenta, kao i broj indeksa studenta, koji odgovarajuće atribute inicijalizira na vrijednosti zadane parametrima, dok se broj položenih ispita i prosječna ocjena inicijaliziraju respektivno na 0 i 5.
c) Trivijalne metode koje vraćaju vrijednosti svih odgovarajućih atributa.
d) Metodu sa jednim parametrom, koja služi za registraciju novog ispita, a koja povećava broj položenih ispita za jedinicu i ažurira prosječnu ocjenu u skladu sa ocjenom iz novopoloženog ispita, koja se zadaje kao parametar (u slučaju neispravne ocjene, treba baciti izuzetak).
e) Metodu koja ispisuje podatke (ime, prezime, broj položenih ispita, prosječna ocjena) o studentu.

Klasa "`Fakultet`" sadrži sljedeće elemente:

a) Privatne atribute koji predstavljaju broj upisanih studenata, maksimalno mogući broj studenata, kao i pokazivač na dinamički alocirani niz pokazivača na objekte tipa "`Student`", koji sadrže podatke u upisanim studentima.
b) Konstruktor sa jednim parametrom, koji vrši dinamičku alokaciju memorije, pri čemu parametar predstavlja maksimalan broj studenata koji se mogu upisati (i koji se ne smije koristiti za potrebe automatske konverzije cjelobrojnih vrijednosti u objekte tipa "`Student`"), kao i odgovarajući destruktor.
c) Konstruktor kopije koji će osigurati da se objekti tipa "`Fakultet`" mogu bezbjedno prenositi kao parametri po vrijednosti, koristiti za inicijalizaciju drugih objekata istog tipa i vraćati kao rezultati iz funkcija.
d) Preklopljeni operator dodjele, koji garantira sigurno dodjeljivanje jednog objekta tipa "`Fakultet`" drugom objektu istog tipa.
e) Metodu koja vrši upis novog studenta, pri čemu se kao parametri metode prenose broj indeksa studenta, kao i njegovo ime i prezime studenta (u slučaju da se dostigne maksimalan broj studenata koji se mogu upisati, treba baciti izuzetak).
f) Metodu koja briše sve upisane studente.
g) Metodu koja vrši registraciju nove ocjene. Metoda kao parametre zahtijeva broj indeksa studenta kao i ocjenu koju je student dobio. Ova metoda treba da ažurira prosječnu ocjenu studenta na osnovu novounesene ocjene.
h) Metodu koja ispisuje podatke (ime, prezime, broj položenih ispita, prosječna ocjena) o studentu sa brojem indeksa koji se zadaje kao parametar.
i) Metodu koja ispisuje imena i prezimena svih studenata čija je prosječna ocjena veća od vrijednosti koja se zadaje kao parametar.

j) Metodu koja ispisuje imena i prezimena svih studenata koji su položili više ispita od broja koji se zadaje kao parametar.
k) Metodu koja ispisuje sve upisane studente sortirane u opadajućem poretku po prosječnoj ocjeni (tj. student sa najvećom prosječnom ocjenom se ispisuje prvi), pri čemu se podaci o svakom studentu ispisuju u posebnom redu.

Implementirajte klase "`Student`" i "`Fakultet`" sa navedenim svojstvima. Sve metode implementirajte izvan tijela klase, osim metoda čija implementacija zahtijeva jednu ili dvije naredbe. Obavezno napišite i mali testni program u kojem će se testirati sve navedene metode.

10. Izmijenite klasu "`Fakultet`" razvijenu u prethodnom zadatku, tako što će se za evidenciju pokazivača na dinamički alocirane objekte tipa "`Student`" umjesto dinamički alociranog niza pokazivača koristiti vektor čiji su elementi pokazivači na objekte tipa "`Student`". Pri tome, atributi koji čuvaju broj upisanih studenata i maksimalno mogući broj studenata više neće biti potrebni. Zaista, broj upisanih studenata se može saznati testiranjem trenutne veličine vektora, dok maksimalno mogući broj studenata više nije potrebno zadavati, s obzirom da vektor može u toku rada po volji povećavati svoju veličinu. Jedina izmjena u interfejsu klase biće u tome što konstruktoru više neće biti potreban parametar (zahvaljujući fleksibilnosti vektora, moći će se upisati onoliko studenata koliko želimo, bez potrebe da unaprijed specificiramo njihov maksimalan broj). Destruktor će i dalje biti potreban (da oslobodi sve dinamički alocirane objekte tipa "`Student`" nakon što objekat tipa "`Fakultet`" prestane postojati), a biće potrebni i konstruktor kopije i preklopljeni operator dodjele, s obzirom da su dinamički alocirani objekti tipa "`Student`" u vlasništvu klase "`Fakultet`", iako nisu njen sastavni dio (razmislite dobro šta konstruktor kopije i preklopljeni operator dodjele tačno trebaju kopirati). Naravno, metoda za upis novog studenta više ne treba provjeravati da li je dostignut maksimalan broj studenata, s obzirom da ograničenje na maksimalan broj studenata više ne postoji.

11*. Za potrebe obrade zahtjeva koji dolaze na studentsku službu od strane studenata, neki računarski program definira i implementira klase nazvane "`Zahtjev`" i "`Zahtjevi`". Klasa "`Zahtjev`" opisuje jedan zahtjev, dok klasa "`Zahtjevi`" predstavlja kolekciju zahtjeva, odnosno objekata tipa "`Zahtjev`".

Klasa "`Zahtjev`" kao atribute sadrži ime i prezime podnosioca zahtjeva, broj indeksa, tekst zahtjeva, kao i atribute koji predstavljaju dan, mjesec i godinu kada je zahtjev podnesen. Pored ovih atributa, klasa "`Zahtjev`" sadrži još samo konstruktor, koji omogućava propisnu inicijalizaciju svih atributa klase na vrijednosti zadane parametrima. Pri tome, konstruktor testora da li je zadan legalan datum (ukoliko nije, potrebno je baciti izuzetak).

Klasa "`Zahtjevi`" predstavlja kolekciju zahtjeva, izvedenu kao dinamički niz pokazivača na objekte tipa "`Zahtjev`". Ova klasa sadrži konstruktor sa jednim parametrom, koji vrši odgovarajuću dinamičku alokaciju memorije, pri čemu parametar predstavlja maksimalan broj zahtjeva koji se mogu pohraniti. Pored toga, klasa sadrži i odgovarajući destruktor, koji oslobađa memoriju koju je objekat tipa "`Zahtjevi`" zauzeo tokom svog života, te konstruktor kopije i preklopljeni operator dodjele koji omogućavaju bezbjedno kopiranje i međusobno dodjeljivanje objekata tipa "`Zahtjevi`". Dalje, ova klasa posjeduje i tri metode. Prva metoda vrši kreiranje i dodavanje novog zahtjeva, i ona ima iste parametre kao i konstruktor klase "`Zahtjev`". Ova metoda kreira novi zahtjev (u skladu sa navedenim parametrima) i smješta ga u kolekciju. Druga metoda nema parametara, i ona vrši ispis podataka o prvom primljenom zahtjevu na ekran (stil ispisa odaberite sami). Pored toga, ova metoda vrši brisanje obrađenog zahtjeva iz kolekcije, tako da će sljedeći poziv ove metode ispisati sljedeći zahtjev po redu, i tako dalje, sve dok se ne istroše svi zahtjevi. U slučaju da se ova metoda pozove kada je kolekcija prazna, metoda treba da baci izuzetak. Konačno, posljednja metoda nema parametara i ona daje kao rezultat logičku vrijednost "`true`" ako i samo ako objekat nad kojim je primijenjena predstavlja praznu kolekciju, a u suprotnom daje kao rezultat logičku vrijednost "`false`".

Napisane klase testirajte u testnom programu koji kreira nekoliko zahtjeva prema podacima unesenim sa tastature, a zatim ispisuje sve unijete zahtjeve u redoslijedu pristizanja.

12. Izmijenite klasu "`Zahtjevi`" razvijenu u prethodnom zadatku, tako što će se za evidenciju pokazivača na dinamički alocirane objekte tipa "`Zahtjev`" umjesto dinamički alociranog niza pokazivača koristiti vektor čiji su elementi pokazivači na objekte tipa "`Zahtjev`". Pri tome, atributi koji čuvaju broj registriranih zahtjeva i maksimalno mogući broj zahtjeva koji se mogu pohraniti više neće biti potrebni. Konstruktor klase više neće imati parametar, dok su destruktor, konstruktor kopije i preklopljeni operator dodjele i dalje potrebni. Metoda za kreiranje novog zahtjeva više ne treba

provjeravati da li je dostignut maksimalan broj zahtjeva, s obzirom da ograničenje na maksimalan broj zahtjeva više ne postoji.

13#. Definirajte i implementirajte klasu "Trougao" (ili "Trokut", ovisno od Vašeg jezičkog opredjeljenja) koja modelira trouglove (trokute) u ravni čija se tjemena nalaze u zadanim tačkama. U nastavku će biti opisano šta ova klasa treba sadržavati. Na prvom mjestu, tu je konstruktor koji omogućava kreiranje trougla na osnovu tri tjemena koji mu se zadaju kao parametri. Tjemena se zadaju kao objekti tipa "Tacka", pri čemu je tip "Tacka" definiran na globalnom nivou pomoću deklaracije

```
typedef std::pair<double, double> Tacka;
```

pri čemu koordinate uređenog para čuvaju Descartesove koordinate tačke. U slučaju da se sva tri tjemena nalaze na jednom pravcu, tako da se od njih ne može formirati trougao, treba baciti izuzetak tipa "domain_error" uz prateći tekst "Nekorektne pozicije tjemena" (za testiranje da li se tri tjemena nalaze na jednom pravcu ili ne, od koristi Vam može biti funkcija "Orijentacija" koju ćemo u nastavku opisati). Za naknadnu izmjenu podataka o trouglu predviđene su dvije verzije metode nazvane "Postavi". Prva verzija prima iste parametre i obavlja isti zadatak kao i konstruktor, samo nad već postojećim objektom, dok druga omogućava da se zada koordinata samo jednog tjemena koje se mijenja, dok ostala dva tjemena ostaju nepromijenjena. Ova verzija ima dva parametra, pri čemu prvi parametar (tipa "int") određuje koje se tjeme mijenja i može imati samo vrijednost 1, 2 ili 3, u suprotnom treba baciti izuzetak tipa "range_error" uz prateći tekst "Nekorektan indeks" (isto vrijedi za sve naredne funkcije koje također traže redni broj tjemena kao parametar).

Za testiranje kako su tri tačke međusobno orijentirane, predviđena je statička funkcija članica nazvana "Orijentacija". Ona prima tri tačke (tj. objekte tipa "Tacka") kao parametre i vraća kao rezultat 1 ukoliko je orijentacija pozitivna (tj. ukoliko je kretanje od prve ka posljednjoj tački u smjeru suprotnom od smjera kazaljke na satu), -1 ukoliko je orijentacija negativna (kretanje u smjeru kazaljke na satu), te 0 ukoliko su tačke kolinearne (tako da nema smisla govoriti o orijentaciji). Ukoliko su (x_1, y_1), (x_2, y_2) i (x_3, y_3) koordinate te tri tačke, test orjentacije se može obaviti testiranjem znaka izraza $x_1 (y_2 - y_3) - x_2 (y_1 - y_3) + x_3 (y_1 - y_2)$ koji je pozitivan za pozitivnu orijentaciju, negativan za negativnu orijentaciju, te nula u slučaju da su razmatrane tačke kolinearne.

Predviđeno je više metoda pomoću kojih se mogu saznavati razne informacije o trouglu. Metoda "DajTjeme" vraća tjeme čiji je redni broj dat kao parametar, metoda "DajStranicu" vraća dužinu stranice koja se nalazi nasuprot tjemena čiji je redni broj dat kao parametar, dok metoda "DajUgao" (dopušteno je i "DajKut") daje iznos ugla trougla u radijanima koji se nalazi u tjemenu čiji je redni broj dat kao parametar (za računanje ugla najlakše je iskoristiti dužine stranica i kosinusnu teoremu, iako postoje i načini zasnovani na analitičkoj geometriji ili vektorskoj algebri direktno iz koordinata tjemena). Parametri sve tri metode su cjelobrojnog tipa, inače se baca izuzetak, isto kao u drugoj verziji funkcije "Postavi". Metoda "DajCentar" (bez parametara) daje kao rezultat tačku koja predstavlja centar odnosno težište trougla (za trougao vrijedi da su koordinate njegovog centra aritmetičke sredine koordinata svih tjemena, ali to inače ne vrijedi generalno za druge poligone). Konačno, metode "DajObim" i "DajPovrsinu" (također bez parametara) daju respektivno obim odnosno površinu trougla. Površina se može izračunati direktno na osnovu koordinata kao jedna polovina apsolutne vrijednosti izraza $x_1 (y_2 - y_3) - x_2 (y_1 - y_3) + x_3 (y_1 - y_2)$. Dobro ste primijetili, ovo je isti izraz kao u funkciji za testiranje orijentacije(ubilo se za pomoćne funkcije).

Metoda "DaLiJePozitivnoOrijentiran" bez parametara treba da vrati logičku vrijednost "tačno" ukoliko je trougao nad kojim je pozvana pozitivno orijentiran (tj. ukoliko su mu tjemena posmatrana u redoslijedu zadavanja pozitivno orijentirana), a "netačno" u suprotnom. Metoda "DaLiJeUnutra" prima tačku kao parametar, a treba da vrati logičku vrijednost "tačno" ako i samo ako se razmatrana tačka nalazi unutar trougla (za tačku koja leži tačno na nekoj od stranica ne smatra se da leži unutar trougla). Mada problem određivanja da li je tačka unutar trougla ili ne izgleda jako težak, on u osnovi to nije. Naime, ukoliko su T_1, T_2 i T_3 koordinate tjemena trougla, a T tačka koja se testira, informacija o tome da li se tačka nalazi unutar trougla ili ne može se dobiti upoređujući orijentaciju trojki tačaka (T_1, T_2, T_3), (T_1, T_2, T), (T_2, T_3, T) i (T_3, T_1, T). Ovim ste dobili osnovnu ideju, a ostatak otkrijte sami (nacrtajte nekoliko crteža i lako ćete izvući zaključke). Podržana je i metoda "Ispisi" bez parametara koja ispisuje na ekran informacije o tjemenima trougla u obliku "$((x_1, y_1), (x_2, y_2), (x_3, y_3))$".

Metoda "`Transliraj`" prima dva realna broja Δx i Δy kao parametre, a obavlja tranlsaciju trougla za Δx jedinica horizontalno i Δy jedinica vertikalno. Predviđen je i specijalan slučaj translacije putem metode "`Centriraj`" koja obavlja takvu translaciju da se nakon nje centar (težište) trougla nađe u tački koja je zadana kao parametar. Metoda "`Rotiraj`" rotira trougao oko zadane tačke (centra rotacije) za zadani ugao. Parametri ove metode su centar rotacije (tipa "`Tacka`"), a drugi je ugao rotacije u radijanima (realan broj). 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 $(x_c + (x - x_c)\cos\alpha - (y - y_c)\sin\alpha, y_c + (x - x_c)\sin\alpha + (y - y_c)\cos\alpha)$. Metoda "`Skaliraj`" skalira trougao tako da mu se sve stranice produže sa zadanim faktorom, ali da trougao ostane i dalje u istom položaju, odnosno da stranice nakon skaliranja ostanu paralelne sa prethodnim stranicama. Prvi parametar ove metode je tzv. *centar skaliranja*, odnosno tačka čije pozicije nakon skaliranja ostaju nepromijenjene. Skaliranje se najlakše obavlja tako što se svaka tačka oblika (x, y) koja opisuje objekat zamjenjuje sa tačkom $(x_c + k\,(x - x_c), y_c + k\,(y - y_c))$ gdje je k faktor skaliranja, koji se zadaje kao drugi parametar (realan broj). Dopušteno je da faktor skaliranja bude i negativan (u tom slučaju skalirani trougao biće zapravo još dodatno zarotiran za 180 °), ali ne i nula (takvo skaliranje bi reduciralo trougao na tačku). Stoga, ukoliko je faktor skaliranja jednak nuli, treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Nekorektan faktor skaliranja". Predviđene su i specijalni slučajevi metoda "`Rotiraj`" i "`Skaliraj`" sa samo jednim parametrom (ugao rotacije odnosno faktor skaliranja), pri čemu se tada za centar rotacije odnosno skaliranja uzima centar (težište) trougla.

Na kraju treba podržati još tri prijateljske funkcije (čija implementacija može biti vrlo jednostavna ako se ispravno odaberu uvjeti). Prijateljska funkcija "`DaLiSuIdenticni`" testira da li se dva trougla (tj. objekta tipa "`Trougao`") koji joj se prenose kao parametri poklapaju, tj. da li su im tjemena na istim mjestima u ravni, a vraća logičku vrijednost "**`true`**" ili "**`false`**", ovisno od rezultata testiranja. Vodite računa da to ne mora značiti da im tjemena sa istim rednim brojem moraju biti na istim mjestima, tako da je recimo trougao sa tjemenima u tačkama $(1, 3)$, $(5, 2)$ i $(4, 4)$ tim redom identičan trouglovima sa tjemenima u tačkama $(5, 2)$, $(1, 3)$ i $(4, 4)$ odnosno $(4, 4)$, $(5, 2)$ i $(1, 3)$ tim redom. Pri tome, obratite pažnju da identični trouglovi ne moraju nužno imati istu orijentaciju (u gore navedenom primjeru, prvi trougao ima pozitivnu, a druga dva negativnu orijentaciju). Funkcija "`DaLiSuPodudarni`" testira da li su dva trougla koji joj se prenose kao parametri podudarni ili ne. Trouglovi su podudarni ukoliko se mogu dovesti na poklapanje translacijom i rotacijom. Na primjer, trougao sa tjemeinima u tačkama $(2, 1)$, $(6, 1)$ i $(6, 4)$ podudaran je sa trouglom sa tjemenima u tačkama $(1, 3)$, $(1, 7)$ i $(4, 3)$. Vodite računa da *nije dovoljno* da dva trougla imaju jednake dužine stranica da bi bili podudarni, nego da su neophodni još neki uvjeti (otkrijte ih sami). Na primjer, trougao sa tjemenima u tačkama $(1, 3)$, $(4, 3)$ i $(4, 7)$ *nije podudaran* sa prethodna dva trougla (ne može se dovesti do poklapanja sa njima samo translacijom i rotacijom, nego je potrebno i *prevrtanje* – nacrtajte sliku), iako sva tri trougla imaju dužine stranica 3, 4 i 5, ne nužno tim redom. Konačno, prijateljska funkcija "`DaLiSuSlicni`" ispituje da li su dva trougla koji joj se prenose kao parametri slična ili ne. Dva trougla su slična ukoliko se mogu dovesti na poklapanje translacijom, rotacijom i skaliranjem. Uvjeti za sličnost su analogni uvjetima za podudarnost, samo što ovdje one stranice koje kod podudarnosti trebaju biti jednake ovdje ne moraju jednake, nego je dovoljno samo da budu međusobno proporcionalne (tj. odnos im treba da bude isti).

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 "`Trougao`". Nakon toga, sa tastature treba redom unositi podatke za n trouglova (podaci o svakom trouglu se unose posebno, redom za prvo, drugo a zatim i za treće tjeme). Za svaki od trouglova, nakon obavljenog unosa treba dinamički kreirati odgovarajući trougao inicijaliziran u skladu sa unesenim podacima i pametni pokazivač na tako kreirani kreirani trougao ubaciti u vektor. Ukoliko korisnik zada tjemena od kojih se ne može formirati trougao, treba ispisati poruku upozorenja i zatražiti novi unos podataka za isti trougao. Nakon okončanja unosa, program treba prvo translirati sve trouglove u skladu sa podacima koji se unose sa tastature, a zatim ih rotirati oko njihovog centra te skalirati uzimajući prvo tjeme kao centar skaliranja, pri čemu se ugao rotacije i faktor skaliranja unose sa tastature. Za tu svrhu trebate 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 trouglove u rastući poredak po površini (tj. trougao sa manjom površinom dolazi prije trougla sa većom površinom), te ispisati podatke o svim trouglovima nakon obavljenog sortiranja. Podaci o svakom trouglu trebaju biti u

posebnom redu. 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. Zatim treba ispisati podatke o trouglu koji ima najmanji obim, za šta ćete iskoristiti funkciju "`min_element`" uz definiranje prikladne funkcije kriterija (ponovo kao lambda funkcije). Konačno, na kraju, program treba pronaći sve parove identičnih, podudarnih i sličnih trouglova i ispisati koji su to trouglovi (ili obavijest da takvih parova nema). Ovo je okvirni opis šta testni program treba da radi, a precizan izgled dijaloga između korisnika i programa biće specificiran putem javnih autotestova.

14#. Definirajte i implementirajte klasu "`NepreklapajuciKrug`" koja omogućava čuvanje podataka koji opisuju jedan krug u ravni, ali uz dodatni uvjet da se ni jedan kreirani krug *ne smije presjecati niti sa jednim drugim krugom* (tj. objektom tipa "`NepreklapajuciKrug`") *koji postoji u isto vrijeme* (ovaj uvjet je zapravo *najteži dio zadatka*, i na kraju postavke će biti data uputa kako ovo postići). Krug je opisan sa tri podatka: x i y koordinatom centra i 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:

```
NepreklapajuciKrug(double r = 0);
NepreklapajuciKrug(double x, double y, double r = 0);
void PostaviPoziciju(double x, double y);
void PostaviPoluprecnik(double r);
double DajX() const;
double DajY() const;
double DajPoluprecnik() const;
double DajObim() const;
double DajPovrsinu() const;
void Ispisi() const;
void Transliraj(double delta_x, double delta_y);
void Rotiraj(double alpha);
void Rotiraj(double alpha, double x_c, double y_c);
```

Konstruktor sa jednim parametrom kreira krug zadanog poluprečnika sa centrom u koordinatnom početku, dok konstruktor sa tri parametra kreira krug sa zadanom lokacijom centra i zadanim poluprečnikom. Oba konstruktora imaju podrazumijevanu vrijednost 0 za poluprečnik, tako da se mogu koristiti i bez parametara, odnosno sa dva parametra. Međutim, ukoliko se krug koji se kreira preklapa sa nekim od krugova koji u tom trenutku već postojii kreiran, baca se izuzetak tipa "`logic_error`" uz prateći tekst "Krug se preklapa sa postojecim krugovima" (kako ovo postići, biće opisano na kraju postavke zadatka). Recimo, ukoliko pokušamo izvršiti kreiranje tri objekta tipa "`NepreklapajuciKrug`" konstrukcijama poput

```
NepreklapajuciKrug k1(9);
NepreklapajuciKrug k2(30, 15, 5);
NepreklapajućiKrug k3(10, 10, 7);
```

treća definicija treba da baci izuzetak, jer se krug sa centrom u tački $(10, 10)$ i poluprečnikom 7 preklapa sa prvim definiranim krugom sa centrom u tački $(0, 0)$ i poluprečnikom 9. Pored toga, konstruktor baca izuzetak tipa tipa "`domain_error`" uz prateći tekst "Ilegalan poluprečnik" ukoliko se kao poluprečnik navede negativan broj. Kopiranje i međusobno dodjeljivanje objekata tipa "`NepreklapajuciKrug`" treba zabraniti (s obzirom da bi se kopija nekog objekta svakako preklapala sa njim samim).

Metode "`PostaviPoziciju`" i "`PostaviPoluprecnik`" omogućavaju naknadnu promjenu pozicije kruga (bez promjene poluprečnika) odnosno poluprečnika (bez promjene pozicije). Pri tome, ove metode će također baciti izuzetak kao i konstruktor ukoliko promjena parametara kruga dovede do njegovog preklapanja sa već postojećim krugovima (pošto će se test na preklapanje izvoditi još na nekoliko mjesta, najbolje ga je izvesti u nekoj privatnoj funkciji članici koju ćete pozivati odakle god zatreba). Metoda "`PostaviPoluprecnik`" također baca izuzetak (poput konstruktora) ukoliko se zada negativan poluprečnik. Metode "`DajX`", "`DajY`", "`DajPoluprecnik`", "`DajObim`" i "`DajPovrsinu`" vraćaju redom x koordinatu centra kruga, y koordinatu centra kruga, poluprečnik, obim i površinu kruga, dok metoda "`Ispisi`" ispisuje na ekran podatke o krugu u obliku "$\{(x, y), r\}$" (nema nikakvih razmaka). Metoda "`Transliraj`" pomjera krug za iznos Δx u smjeru x-ose i iznos Δy u smjeru y-ose, pri čemu se vrijednosti Δx i Δy navode kao parametri. 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 zadaju i koordinave tačke (x_c, y_c) oko koje

se vrši rotacija. Za one koji nisu dovoljno upućeni u analitičku geometriju, napomenimo da se rotacijom tačke (x, y) oko tačke (x_c, y_c) za ugao α dobija tačka (x', y') gdje su x' i y' dati kao $x' = x_c + (x - x_c)\cos\alpha - (y - y_c)\sin\alpha$ i $y' = y_c + (x - x_c)\sin\alpha + (y - y_c)\cos\alpha$. Ugao α se zadaje *u radijanima*. Naravno, prilikom translacije i rotacije, mijenja se samo pozicija kruga, dok njegov poluprečnik ostaje isti. Ove dvije metode također bacaju izuzetak ukoliko pomjeranje kruga dovede do njegovog preklapanja sa već postojećim krugovima.

Napisanu klasu demonstrirajte u testnom programu koji traži da se tastature unese prirodan broj n, koji zatim treba dinamiički alocirati niz od n pokazivača na objekte tipa "`NepreklapajuciKrug`" koje treba dinamički alocirati i inicijalizirati na osnovu podataka koji se unose sa tastature (podaci o svakom krugu se unose posebno), vodeći računa da kreiranje kruga može i da baci izuzetak. Nakon okončanja unosa, program treba prvo translirati a zatim rotirati sve unesene krugove 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. Pri tome, kako se krugovi transformiraju jedan po jedan, može se desiti da prilikom translacije ili rotacije dođe do privremenog preklapanja nekih krugova (iako se na kraju oni opet ne bi preklapali), što može dovesti do bacanja izuzetka. U tom slučaju, prosto treba prijaviti problem pri transformiranju i obustaviti dalji rad programa (taj problem privremenih preklapanja se teško može riješiti bez uvođenja novih funkcionalnosti u interfejs klase, osim tehnikom tzv. inkrementalnog pomjeranja, koja je dosta složena i neefikasna). Nakon obavljenih transformacija, treba ispisati podatke o svim unesenim krugovima nakon obavljenih transformacija. Podaci za svaki krug trebaju biti u posebnom redu. Ispis izvršite uz pomoć funkcije "`foreach`" i prikladne lambda funkcije. Na kraju, program treba ispisati podatke o krugu koji ima najveću površinu, za šta ćete iskoristiti funkciju "`max_element`" uz definiranje prikladne funkcije kriterija (ponovo kao lambda funkcije).

Uputa: Najveći problem je kako detektirati da li se krug koji kreiramo preklapa sa krugovima koji su već kreirani, odnosno kako konstruktor kruga koji 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 bolje sa aspekta utroška resursa (koje trebate ovdje izvesti). Naime, 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 dijeljeni statički atribut koji će sadržavati pokazivač na posljednji kreirani krug (ili nul-pokazivač ukoliko niti jedan krug nije kreiran). Ovaj atribut je potreban da bismo znali gdje počinje "lanac" povezanih krugova. Sad se test na preklapanje izvodi tako što se prođe kroz čitavu listu i testira preklapanje kruga koji 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!

15#. Napravite klasu "`GradjaninBiH`" koja modelira jednog građanina Bosne i Hercegovine (misli se na građanina u pravnom smislu kao stanovnika države, a ne nužno osobu koja živi u gradu). Za kreiranje objekata tipa "`GradjaninBiH`" predviđena su dva konstruktora Oba konstruktora kao prvi parametar prihvataju ime građanina (sa prezimenom), koje se zadaje kao objekat tipa "`string`". Prvi konstruktor, kao drugi parametar zahtijeva jedinstveni matični broj građanina (JMBG), koji je cijeli broj tipa "`long long int`". Inače, JMBG je 13-cifreni broj koji sadrži vrlo mnogo informacija o građaninu. Ako predstavimo JMBG u obliku $c_1c_2c_3c_4c_5c_6c_7c_8c_9c_{10}c_{11}c_{12}c_{13}$, tada c_1c_2 predstavlja dan rođenja, c_3c_4 mjesec rođenja, $c_5c_6c_7$ posljednje 3 cifre godine rođenja. c_8c_9 predstavljaju šifru regije rođenja (npr. 17 za Sarajevo), $c_{10}c_{11}c_{12}$ je kôd po kojem se međusobno razlikuju osobe rođene na isti dan u istoj regiji, pri čemu su za muške osobe rezervirani kodovi od 000 do 499, a za ženske osobe od 500 do 999, dok je c_{13} kontrolna cifra koja zavisi od ostalih 12 cifara prema formuli

$$c_{13} = 11 - (7\cdot(c_1 + c_7) + 6\cdot(c_2 + c_8) + 5\cdot(c_3 + c_9) + 4\cdot(c_4 + c_{10}) + 3\cdot(c_5 + c_{11}) + 2\cdot(c_6 + c_{12}))\,\%\,11$$

pri čemu se ukoliko se dobije $c_{13} = 11$, uzima se da je $c_{13} = 0$, a ako se dobije $c_{13} = 10$, smatra se da je kôd osobe neprikladan (pri dodjeljivanju JMBG tada se bira novi kôd). U slučaju da se zada JMBG koji nije validan, treba baciti izuzetak tipa "`logic_error`" uz prateći tekst "JMBG nije validan". JMBG nije validan ukoliko prvih 7 cifara ne mogu formirati ispravan datum, ili ukoliko se posljednja cifra razlikuje od onoga što bi se dobilo gore prikazanom formulom, što uključuje i slučaj za koji bi formula dala da treba biti $c_{13} = 10$. Ukoliko se zadani JMBG poklapa sa JMBG nekog drugog građanina (tj. nekog drugog objekta tipa "`GradjaninBiH`"), treba baciti izuzetak tipa "`logic_error`" uz prateći tekst "Vec postoji gradjanin sa istim JMBG". Na primjer, ukoliko pokušamo izvršiti deklaracije

```
GradjaninBiH g1("Rambo Sulejmanovic", 1305956174235);
GradjaninBiH g2("Zan Klod Sejdic", 1305956174235);
```

druga deklaracija treba da baci izuzetak, zbog toga što već postoji građanin sa istim JMBG. Kopiranje i međusobno dodjeljivanje objekata ovog tipa treba zabraniti (s obzirom da bi kopija nekog objekta tipa "`GradjaninBiH`" svakako imala isti JMBG). Kako riješiti problem da li postoji neki drugi objekta tipa "`GradjaninBiH`" koji ima isti JMBG biće uskoro opisano.

Drugi konstruktor omogućava da se podaci o građaninu daju putem dana, mjeseca i godine rođenja, šifre regije rođenja, te pola, koji se zadaji kao parametri (navedenim redom). Svi parametri su cijeli brojevi (tipa "`int`"), osim parametra kojim se zadaje pol koji je tipa "`Pol`", pri čemu je "`Pol`" pobrojani tip definiran u javnom dijelu klase deklaracijom

```
enum Pol {Musko, Zensko};
```

Ukoliko se pokušaju zadati neispravni podaci, treba baciti izuzetak tipa "`logic_error`" uz prateći tekst "Neispravni podaci". Podaci su neispravni ukoliko dan, mjesec i godina rođenja nemaju smisla, ili ako šifra regije nije u opsegu od 0 do 99 (nisu ni sve šifre u ovom opsegu validne, ali tu činjenicu ćemo ovdje ignorirati). Ovaj konstruktor također na osnovu zadanih podataka treba da automatski kreira JMBG. Svi podaci za tu svrhu su poznati, osim kôda osobe. Za tu svrhu, treba uzeti prvi slobodni kôd, tj. najmanji kôd koji dosad nije iskorišten ni za jednu drugu osobu istog pola rođenu na isti dan u istoj regiji. Kako pronaći JMBG drugih građanina (tj. drugih već kreiranih objekata tipa "`GradjaninBiH`") biće također uskoro opisano.

Pored konstruktora, klasa "`GradjaninBiH`" treba da podržava pristupne funkcije "`DajImeIPrezime`", "`DajJMBG`", "`DajDanRodjenja`", "`DajMjesecRodjenja`", "`DajGodinuRodjenja`", "`DajSifruRegije`" i "`DajPol`" koje vraćaju informacije koje su jasne iz imena funkcija (sve ove funkcije su bez parametara). Pri tome, treba imati na umu ukoliko je građanin zadan putem JMBG, nije moguće jednoznačno odrediti godinu rođenja, jer prva cifra godine rođenja nije sadržana u JMBG. Zbog toga, za realizaciju funkcije koja daje godinu rođenja uzeti pretpostavku da nijedan građanin nije stariji od 100 godina, kao i da je trenutno 2021. godina. Pored ovih pristupnih funkcija, klasa treba da podržava još jedino funkciju "`PromijeniImeIPrezime`", koja omogućava promjenu imena građanina, a korisna je ukoliko recimo Brus Li Ramadanović poželi da promijeni ime i prezime u Čak Noris Bičakčić.

Kao i u prethodnom zadatku, najveći problem u ovom zadatku je kako detektirati da li građanin kojeg kreiramo ima isti JMBG kao građani koji su već kreirani, odnosno kako konstruktor objekta tipa "`GradjaninBiH`" kojeg kreiramo može znati za druge objekte istog tipa. Problem se može riješiti koristeći posve iste ideje koje su objašnjene u prethodom zadatku.

Obavezno napišite i kratak testni program u kojem ćete testirati napisanu klasu (eventualni dijalozi između korisnika i programa biće specificirani javnim autotestovima). Za realizaciju programa najstrože je zabranjeno koristiti bibliotečke kontejnerske tipove podataka, kao što su vektori, dekovi, liste, skupovi, mape, itd.

16#.Date su sljedeće deklaracije na globalnom nivou:

```
typedef std::pair<double, double> Tacka;
enum Pozicija {GoreLijevo, GoreDesno, DoljeLijevo, DoljeDesno};
enum Smjer {Nalijevo, Nadesno};
```

Definirajte i implementirajte klasu "`Pravougaonik`" koja modelira pravougaonike u ravni čije su stranice paralelne koordinatnim osama. Klasa treba da ima sljedeći interfejs:

```
Pravougaonik(const Tacka &t1, const Tacka &t2);
void Postavi(const Tacka &t1, const Tacka &t2);
void Postavi(Pozicija p, const Tacka &t);
void Centriraj(const Tacka &t);
Tacka DajTjeme(Pozicija p) const;
Tacka DajCentar() const;
double DajHorizontalnu() const;
double DajVertikalnu() const;
double DajObim() const;
double DajPovrsinu() const;
static Pravougaonik Presjek(const Pravougaonik &p1, const Pravougaonik &p2);
void Transliraj(double delta_x, double delta_y);
void Rotiraj(const Tacka &t, Smjer s);
void Ispisi() const;
friend bool DaLiSePoklapaju(const Pravougaonik &p1, const Pravougaonik &p2);
friend bool DaLiSuPodudarni(const Pravougaonik &p1, const Pravougaonik &p2);
friend bool DaLiSuSlicni(const Pravougaonik &p1, const Pravougaonik &p2);
```

Konstruktor omogućava kreiranje pravougaonika na osnovu koordinata dva nasuprotna tjemena (vrha) pravougaonika, koje se nalaze respektivno u parametrima "`t1`" i "`t2`" (s obzirom da su stranice paralelne koordinatnim osama, pozicije ova dva tjemena jednoznačno određuju pravougaonik). Pri tome nije specificirano koja tačno tjemena "`t1`" i "`t2`" predstavljaju, jedino je poznato da su to dva nasuprotna tjemena (recimo "`t1`" može biti gornji desni, a "`t2`" donji lijevi vrh). Dozvoljeno je kreirati i pravougaonike koji se degeneriraju u duž, pa čak i u tačku (to će se desiti kada su "`t1`" i "`t2`" dvije iste tačke). Za naknadnu izmjenu podataka o pravougaoniku predviđene su dvije verzije metode "`Postavi`". Prva prima iste parametre i obavlja isti zadatak kao i konstruktor, samo nad već postojećim objektom, dok druga omogućava da se zada koordinata samo jednog tjemena koje se mijenja, dok njemu nasuprotno tjeme ostaje na istoj poziciji. Koje se tjeme mijenja, zadaje se parametrom "`p`" (koji može imati vrijednosti "`GoreLijevo`", "`GoreDesno`", "`DoljeLijevo`" i "`DoljeDesno`"), dok parametar "`t`" predstavlja novu poziciju tjemena. Predviđena je i funkcija "`Centriraj`", nakon čijeg poziva centar pravougaonika treba da se nađe u zadanoj tački, dok mu dužine stranica i orijentacija ostaju nepromijenjeni.

Funkcija "`DajTjeme`" omogućava da se sazna pozicija svakog od 4 moguća tjemena pravougaonika, pri čemu preko parametra "`p`" zadajemo koje nas tjeme zanima, dok funkcija "`DajCentar`" vraća informacije o koordinatama centra pravougaonika. Funkcije "`DajHorizontalnu`" i "`DajVertikalnu`" vraćaju dužine horizontalne i vertikalne stranice pravougaonika respektivno, dok funkcije "`DajObim`" i "`DajPovrsinu`" vraćaju respektivno njegov obim i površinu (za pravougaonike koji se degeneriraju na duž ili tačku, moguće je da dužina neke od stranica bude 0).

Statička funkcija članica "`Presjek`" prima kao parametre dva objekta tipa "`Pravougaonik`", a kao rezultat vraća novi objekat tipa "`Pravougaonik`" koji predstavlja presjek (tj. skup svih zajedničkih tačaka) pravougaonika koji su preneseni kao parametri. Naime, presjek dva pravougaonika, ukoliko uopće postoji, uvijek je također pravougaonik (pretpostavlja se da pravougaonik čine i sve tačke u njegovoj unutrašnjosti). Ukoliko pravougaonici preneseni kao parametar nemaju zajedničkih tačaka, funkcija baca izuzetak tipa "`domain_error`" uz prateći tekst "Pravougaonici se ne presijecaju". Dozvoljeno je da presjek bude i pravougaonik koji se degenerira na duž ili tačku.

Funkcija "`Transliraj`" vrši tranlsaciju pravougaonika za "`delta_x`" jedinica horizontalno i "`delta_y`" jedinica vertikalno, dok funkcija "`Rotiraj`" rotira pravougaonik za 90° oko tačke čije su koordinate zadane parametrom "`t`" nalijevo (u smjeru suprotnom od kazaljke na satu) ili nadesno (u smjeru kazaljke na satu) ovisno od toga kakav je parametar "`s`". Ugao rotacije je ograničen isključivo na 90° da bi stranice pravougaonika i dalje ostale paralelne koordinatnim osama. Funkcija "`Ispisi`" ispisuje podatke o pravougaoniku u formatu "$\{\{x',y'\},\{x'',y''\}\}$", pri čemu su (x', y') i (x'', y'') koordinate gornjeg lijevog odnosno donjeg desnog ugla respektivno.

Prijateljska funkcija "`DaLiSePoklapaju`" testira da li se pravougaonici koji joj se prenose kao parametri poklapaju, tj. da li su im vrhovi na istim mjestima u ravni, i vraća logičku vrijednost "`true`" ili "`false`", ovisno od rezultata testiranja. Vodite računa da se pravougaonik zadaje preko *samo dva vrha*, tako da se na primjer pravougaonik zadan vrhovima $(1, 1)$ i $(7, 5)$ poklapa sa pravougaonikom zadanim vrhovima $(7, 1)$ i $(1, 5)$. Prijateljska funkcija "`DaLiSuPodudarni`" testira da li su pravougaonici koji se prenose kao parametri podudarni ili ne. Pravougaonici su podudarni ukoliko se mogu dovesti na poklapanje translacijom i rotacijom, što se u suštini svodi na to da su im

stranice istih dužina, neovisno od orjentacije (tako da recimo horizontalna stranica jednog pravougaonika može biti jednaka vertikalnoj stranici drugog i obrnuto). Konačno, prijateljska funkcija "`DaLiSuSlicni`" ispituje da li su dva pravougaonika slična ili ne. Dva pravougaonika su slična ukoliko su im stranice međusobno proporcionalne.

Napisanu klasu demonstrirajte u testnom programu koji traži da se tastature unese prirodan broj n, koji zatim treba dinamički alocirati niz od n pokazivača na objekte tipa "`Pravougaonik`" koje treba dinamički alocirati i inicijalizirati na osnovu podataka koji se unose sa tastature (podaci o svakom pravougaoniku se unose posebno, prvo za jedno, a zatim za drugo tjeme). Nakon okončanja unosa, program treba prvo translirati sve pravougaonike u skladu sa podacima koji se unose sa tastature, a zatim ih rotirati oko njihovog centra. 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 ispisati podatke o svim unesenim pravougaonicima nakon obavljenih transformacija. Podaci za svaki pravougaonik trebaju biti u posebnom redu. Ispis izvršite uz pomoć funkcije "`foreach`" i prikladne lambda funkcije. Na kraju, program treba ispisati podatke o pravougaoniku koji ima najveću površinu, za šta ćete iskoristiti funkciju "`max_element`" uz definiranje prikladne funkcije kriterija (ponovo kao lambda funkcije).

Uputa: Vjerovatno će Vam najteže biti naći presjek dva pravougaonika, jer izgleda da treba ispitati jako mnogo situacija koje mogu nastupiti. Međutim, to uopće nije tako teško kako izgleda. Naime, neki pravougaonik čiji gornji lijevi ugao odnosno donji desni ugao imaju koordinate (x', y') i (x'', y'') respektivno, može se prikazati kao presjek četiri poluravni $x \geq x'$, $x \leq x''$, $y \leq y'$ i $y \geq y'$. Stoga, da bismo presjekli neki pravougaonik s ovim pravougaonikom, dovoljno ga je presjeći redom s ove četiri poluravni, što je mnogo lakše uraditi (svako od ova četiri presijecanja može eventualno promijeniti poziciju samo jednog od dva nasuprotna tjemena).

17#. Prepravite klasu "`Pravougaonik`" iz prethodnog zadatka u klasu "`NepreklapajuciPravougaonik`" koja ima praktično identičan interfejs kao i prethodna klasa, uz jedine razlike u interfejsu što je svuda "`Pravougaonik`" zamijenjeno sa "`NepreklapajuciPravougaonik`" i što je izbačena funkcija "`Presjek`". Objekti tipa "`NepreklapajuciPravougaonik`" u suštini se razlikuju od objekata tipa "`Pravougaonik`" 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). Za dva pravougaonika smatramo da se preklapaju ukoliko im presjek nije prazan skup i ukoliko im se presjek ne degenerira u duž ili u tačku (tj. ukoliko im presjek ima nenultu površinu). 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 "`NepreklapajuciPravougaonik`" konstrukcijama poput

```
NepreklapajuciPravougaonik p1({3, 2}, {7, 4});
NepreklapajuciPravougaonik p2({10, 4}, {8, 1});
NepreklapajuciPravougaonik p3({6, 1}, {4, 5});
```

treća definicija treba da baci izuzetak, jer se pravougaonik sa nasuprotnim tjemenima u tačkama $(6, 1)$ i $(4, 5)$ preklapa sa ranije definiranim pravougaonikom "`p1`" sa nasuprotnim tjemenima u tačkama $(3, 2)$ i $(7, 4)$. Kopiranje i međusobno dodjeljivanje objekata ovog tipa treba zabraniti (s obzirom da bi se kopija nekog objekta tipa "`NepreklapajuciPravougaonik`" svakako preklapala sa njim samim). Također, očigledno je da postaje besmislena i funkcija "`Presjek`", s obzirom da dva objekta ovog tipa po samom načinu njihove konstrukcije ne mogu imati presjek.

Sve funkcije inspektori u klasi "`NepreklapajuciPravougaonik`" ostaju posve identične kao u klasi "`Pravougaonik`". Međutim, sve funkcije mutatori trebaju baciti isti izuzetak kao i konstruktor ukoliko promjena parametara pravougaonika dovede do njegovog preklapanja sa drugim već postojećim pravougaonicima (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 pravougaonike (tj. na objekte tipa "`NepreklapajuciPravougaonik`"). Nakon toga, sa tastature treba redom unositi podatke za n pravougaonika (na isti način kao u prethodnom zadatku). Za svaki pravougaonik, nakon obavljenog unosa treba dinamički kreirati odgovarajući pravougaonik inicijaliziran u skladu sa unesenim

podacima i pametni pokazivač na tako kreirani pravougaonik ubaciti u vektor. Ukoliko konstrukcija ne uspije jer se uneseni pravougaonik preklapa sa do tada unesenim pravougaonicima, treba ispisati poruku upozorenja i zatražiti novi unos podataka za isti pravougaonik. Nakon okončanja unosa, program treba sortirati sve unesene pravougaonike u rastući poredak po površini (tj. pravougaonik sa manjom površinom dolazi prije pravougaonika sa većom površinom) i ispisati podatke o svim pravougaonicima nakon obavljenog sortiranja. Za sortiranje obavezno koristiti bibliotečku funkciju "`sort`" uz pogodno definiranu funkciju kriterija kao lambda funkciju. Na kraju, program treba pronaći sve parove podudarnih pravougaonika i ispisati koji su to pravougaonici (ili obavijest da takvih parova nema).

Uputa: Najveći problem je kako detektirati da li se pravougaonik koji kreiramo preklapa sa pravougaonicima koji su već kreirani, odnosno kako konstruktor pravougaonika koji kreiramo može znati za druge pravougaonike. To možete i trebate izvesti na isti način kako je opisano u uputama na kraju Zadatka 14.

18#. Radi učestalih nedolazaka na posao i izbjegavanja radnih obaveza, rukovodsto Elektrotehničkog fakulteta u Sarajevu odlučilo je da uvede novi sistem evidentiranja dolaska i odlaska zaposlenika na posao. Pošto je novi sistem registriranja veoma zahtjevan, zabrinutoj administrativnoj službi je potreban računarski program koji će voditi evidenciju o broju sati koji su pojedini zaposlenici proveli na poslu u toku radne sedmice, i na osnovu toga obračunati njihovu sedmičnu platu i naknadu za topli obrok.

Fakultet radi od ponedjeljka do petka. Normalno radno vrijeme je od 8:00 do 16:00, ali se pojedinim zaposlenicima, u zavisnosti od složenosti posla koji obavljaju (recimo, nastavnom osoblju) može dopustiti manje sedmično opterećenje, bez umanjenja plate za tu sedmicu. Stoga, za svakog zaposlenika postoje individualni podaci o njegovoj nominalnoj sedmičnoj plati kao i minimalnom sedmičnom opterećenju koje zaposlenik mora provesti na poslu da bi zaslužio puni iznos plate za tu sedmicu (ovo opterećenje ne smije biti veće od Zakonom predviđenog sedmičnog opterećenja od 40 sati). Zaposlenicima koji provedu manji broj sati u toku jedne sedmice od minimalnog sedmičnog opterećenja, odbija se za svaki puni sat ispod minimalne norme iznos jedne satnice, koja se dobija kao iznos sedmične plate podijeljen sa tipičnim sedmičnim opterećenjem od 40 sati. Pored toga, svi zaposlenici imaju pravo na dnevnu naknadu za topli obrok, koja je fiksna za sve zaposlenike (tj. ne zavisi od njihove plate). Zaposlenik stiče pravo na dnevnu naknadu za topli obrok za one dane u toku kojih je na poslu proveo veći broj sati od predviđene kvote, koja je također fiksna za sve zaposlenike. Na kraju, predviđena je i mogućnost dodatnog plaćanja prekovremenog rada. Ukoliko neki zaposlenik provede u toku jednog dana više od 8 sati na poslu, za svaki sat prekovremenog rada u toku tog dana isplaćuje mu se naknada u iznosu od 1.5 satnice za svaki sat prekovremenog rada. Treba napomenuti da se fakultet zaključava u 20:00, tako da do tog vremena svi zaposlenici moraju napustiti fakultet.

Za opisanu svrhu potrebno je razviti program koji će se zasnivati na dvije klase nazvane "`Zaposlenik`" i "`Fakultet`". Klasa "`Zaposlenik`" čuva podatke o jednom zaposleniku, dok je "`Fakultet`" kontejnerska klasa koja čuva podatke o kolekciji objekata tipa "`Zaposlenik`". Ona treba da sadrži sljedeće elemente:

a) Privatne atribute koji predstavljaju broj zaposlenika na fakultetu, maksimalan broj zaposlenika koji fakultet može zaposliti, dnevnu naknadu za topli obrok i minimalnu satnicu neophodnu za ostvarivanje prava na topli obrok (s obzirom da su ovi podaci isti za sve zaposlenike, njih nije neophodno čuvati za svakog zaposlenika posebno), kao i atribut koji služi za pristup dinamički alociranom nizu koji čuva pokazivače na objekte tipa "`Zaposlenik`".
b) Konstruktor sa tri parametra, koji predstavljaju maksimalnan broj zaposlenika na fakultetu, dnevnu naknadu za topli obrok i minimalnu satnicu neophodnu za ostvarivanje prava na topli obrok. Ovaj konstruktor treba inicijalizirati odgovarajuće atribute i izvršiti neophodnu alokaciju prostora za čuvanje podataka o zaposlenicima. Broj zaposlenika se inicijalizira na 0, s obzirom da će se informacije o zaposlenicima dodavati naknadno.
c) Destruktor, koji oslobađa sve resurse koji je objekat tipa "`Fakultet`" zauzeo tokom svog postojanja.
d) Konstruktor kopije i preklopljeni operator dodjele koji omogućavaju sigurno kopiranje i međusobno dodjeljivanje objekata tipa "`Fakultet`" (što se tiče preklopljenog operatora dodjele, najbolje je prvo uništiti sadržaj odredišnog objekta a zatim iskopirati izvorni u odredišni objekat, s obzirom da bi svako drugo rješenje koje bi eventualno iskoristilo već postojeći prostor u odredišnom objektu bilo dosta komplicirano).

e) Metodu koja kreira i evidentira novog zaposlenika. Parametri ove metode su evidencijski platni broj zaposlenika (taj broj jednoznačno određuje zaposlenika), njegovo ime i prezime, iznos njegove sedmične plate, kao i minimalno sedmično opterećenje koje zaposlenik treba provesti na poslu da bi zaradio puni iznos plate (ovi podaci se mogu razlikovati od jednog do drugog zaposlenika). Metoda treba da baci izuzetak ukoliko već postoji zaposlenik sa zadanim platnim brojem, ili ukoliko je dostignut maksimalni broj zaposlenika koji fakultet može primiti.

f) Metodu koja vrši evidenciju dolaska zaposlenika na posao. Parametri metode su platni broj zaposlenika, kao i sat i minuta kada je zaposlenik došao na posao. Metoda treba da baci izuzetak ukoliko zaposlenik sa zadanim platnim brojem ne postoji, ukoliko vrijeme dolaska nije u intervalu od 8:00 do 16:00, ili ukoliko je njegov dolazak za taj dan već evidentiran (novo evidentiranje je moguće tek nakon evidencije odlaska).

g) Metodu koja vrši evidenciju odlaska zaposlenika sa posla. Parametri su isti kao kod prethodne metode. Metoda treba da baci izuzetak ukoliko zaposlenik sa zadanim platnim brojem ne postoji, ukoliko vrijeme odlaska nije u intervalu od 8:00 do 20:00, ukoliko je vrijeme odlaska manje od vremena dolaska za taj dan, ili ukoliko je odlazak zaposlenika za taj dan već evidentiran (novo odjavljivanje je moguće tek nakon nove evidencije dolaska).

h) Metodu koja vrši ispis podataka o zaposleniku. Parametar metode je platni broj zaposlenika, a metoda treba da ispiše ime zaposlenika, platni broj, broj normalno provedenih radnih sati (od početka evidentiranja), broj prekovremenih radnih sati (također od početka evidentiranja), platu za normalne radne sate (u skladu sa trenutnim stanjem, koje će biti kompletno tek na kraju sedmice), dodatak za prekovremeni rad, naknadu za topli obrok i ukupan iznos za isplatu (sve u skladu sa trenutnim stanjem).

i) Metodu koja vrši ispis platne liste za sve zaposlenike na fakultetu (ova metoda bi se tipično trebala pozivati na kraju svake radne sedmice, jer će tek tada svi podaci biti kompletirani). Metoda treba za svakog zaposlenika ispisati iste podatke kao u predhodnoj metodi.

j) Metodu koja vrši zaključivanje radne sedmice. Nakon poziva ove metode, sve informacije o broju provedenih radnih sati za sve zaposlenike vraćaju se na nulu, čime je sistem spreman za novo evidentiranje u toku sljedeće sedmice.

U gore navedenim specifikacijama nije ništa rečeno o tome šta treba sadržavati klasa "`Zaposlenik`". Kako je ona samo pomoćna klasa neophodna za funkcioniranje klase "`Fakultet`" (koja se na nju oslanja), nju možete dizajnirati kako god želite, pod uvjetom da to obezbijedi ispravno funkcioniranje klase "`Fakultet`" u skladu sa gore postavljenim zahtjevima. Sve metode implementirajte izvan tijela klase, osim metoda čija implementacija zahtijeva jednu ili dvije naredbe. Obavezno napišite i mali testni program u kojem će se testirati sve navedene metode. Najbolje bi bilo da program manipulira sa podacima koji se unose sa tastature.

19#. Za potrebe neke aplikacije koja se bavi računarskom grafikom, potrebno je razviti klasu "`Krugovi`" koja čuva kolekciju međusobno nepreklapajućih krugova. Podaci o krugovima čuvaju se u dinamički alociranom nizu objekata tipa "`Krug`", kojem se pristupa putem nekog od atributa klase "`Krugovi`". Pri tome je "`Krug`" struktura definirana kao

```
struct Krug { double x, y, r; };
```

Atributi "`x`", "`y`" i "`r`" u objektu tipa "`Krug`" predstavljaju redom x i y koordinatu centra kruga, te njegov poluprečnik. Interfejs klase "`Krugovi`" treba sadržavati sljedeće elemente:

a) Konstruktor sa jednim parametrom, koji vrši dinamičku alokaciju prostora za smještanje podataka o krugovima. Parametar predstavlja maksimalni broj krugova koji se mogu smjestiti. Ovaj konstruktor se ne smije koristiti za automatsku konverziju cijelih brojeva u objekte tipa "`Krugovi`".

b) Destruktor, koji oslobađa resurse koje su primjerci klase "`Krugovi`" zauzeli tokom svog života.

c) Konstruktor kopije i preklopljeni operator dodjele koji omogućavaju da se objekti tipa "`Krugovi`" mogu bezbjedno kopirati i međusobno dodjeljvati koristeći strategiju dubokog kopiranja.

d) Metodu koja dodaje novi krug u kolekciju. Parametri su koordinate centra kruga i njegov poluprečnik. Pri tome se novododani krug ne smije preklapati niti sa jednim od krugova koji se već nalaze u kolekciji. Ukoliko se to desi, treba baciti izuzetak. Izuzetak se također treba baciti i u slučaju da je kolekcija već popunjena. Napomena: dva kruga se preklapaju ako i samo ako je zbir njihovih poluprečnika veći od rastojanja njihovih centara. Rastojanje između dvije tačke (x_1, y_1) i (x_2, y_2) može se izračunati po formuli $d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

e) Metodu koji vraća referencu na i-ti krug pohranjen u kolekciji, gdje se indeks i zadaje kao parametar. Potrebno je podržati i konstantnu verzuju ove metode, koja u slučaju da se metoda pozove nad konstantnim objektom, vrati kao rezultat kopiju kruga iz kolekcije a ne referencu.

f) Metodu koja ispisuje podatke o svim krugovima na ekran, sortirane u opadajući redoslijed po površini krugova (tj. krug sa najvećom površinom prikazuje se prvi). Podaci o krugu čiji je centar u tački (x, y) a poluprečnik r prikazuju se u formi "$[(x,y),r]$".

20*. Definirajte i implementirajte klasu "`Poligon`" koja predstavlja poligon odnosno mnogougao (ili mnogokut) u ravni. Poligon je opisan koordinatama svojih tjemena, koje se čuvaju u dva dinamički alocirana niza realnih brojeva (jedan čuva x a drugi y koordinate tjemena), kojima se pristupa preko odgovarajućih pokazivača koje predstavljaju atribute klase. Implementacije svih metoda treba izvesti izvan klase. Interfejs klase sadrži sljedeće elemente:

a) Konstruktor sa jednim parametrom, koji predstavlja broj tjemena poligona. Ovaj konstruktor, između ostalog, vrši dinamičku alokaciju prostora za pamćenje koordinata tjemena. Pored toga, koordinate svih tjemena novokreiranog poligona treba postaviti na nulu. Ne smije se dozvoliti da se preko ovog konstruktora može ostvariti automatska konverzija cijelih brojeva u objekte tipa "`Poligon`". Ukoliko je broj tjemena manji od 3, treba baciti izuzetak (jer poligoni sa manje od 3 tjemena ne postoje).

b) Destruktor, koji oslobađa svu memoriju koju je objekat zauzeo tokom svog života.

c) Konstruktor kopije i preklopljeni operator dodjele koji omogućavaju bezbjedno kopiranje i međusobno dodjeljivanje objekata tipa "`Poligon`" zasnovano na strategiji dubokog kopiranja.

d) Metodu koja daje broj tjemena poligona.

e) Metodu koja omogućava zadavanje koordinata tjemena poligona. Parametri su redni broj tjemena (u opsegu od 1 do n gdje je n ukupan broj tjemena), te x i y koordinata tjemena. Ukoliko redni broj tjemena nije ispravan, treba baciti izuzetak (vodite računa da indeksi nizova idu od nule).

f) Metodu koja omogućava saznavanje informacija o i-tom tjemenu poligona, pri čemu se i zadaje kao parametar. Ovaj operator kao rezultat daje objekat tipa "`Tjeme`", koji je definiran kao jednostavna struktura sa dva atributa "`x`" i "`y`" koji predstavljaju x i y koordinatu tjemena respektivno (pretpostavite da je takva struktura definirana negdje u programu).

g) Metodu koja daje kao rezultat površinu poligona. Površina poligona sa n tjemena čije su koordinate (x_i, y_i), $i = 1..n$ računa se po formuli

$$\mathrm{P} = \frac{1}{2}\left| x_n\, y_1 - x_1\, y_n + \sum_{i=1}^{n-1} (x_i\, y_{i+1} - x_{i+1}\, y_i) \right|$$

h) Metodu koja daje kao rezultat informaciju da li je poligon degeneriran ili nije. Poligon je degeneriran ukoliko mu makar jedan par susjednih tjemena ima jednake koordinate, inače nije.

21#. Potrebno je napraviti program koji vrši najavu letova na aerodromskom displeju. Program treba najavljivati sve letove u toku dana, kao i eventualna kašnjenja u polijetanjima. Za tu svrhu, u programu je potrebno razviti dvije klase nazvane "`Let`" i "`Letovi`". Klasa "`Let`" vodi evidenciju o jednom letu, dok klasa "`Letovi`" vodi evidenciju o svim letovima u toku dana. Klasa "`Let`" sadrži privatne atribute koji redom predstavljaju naziv odredišta, oznaku leta (npr. "JFK 327"), broj izlazne kapije (cijeli broj), vrijeme polijetanja (sati i minute), trajanje leta u minutama, kao i informaciju o eventualnom kašnjenju (također u minutama). Naziv odredišta i oznaka leta čuvaju se kao nizovi znakova (ne kao dinamički stringovi), sa maksimalnim dužinama od 30 i 10 znakova respektivno. Ove maksimalne dužine treba definirati kao statičke konstantne atribute. Interfejs klase "`Let`" je sljedeći:

```
Let(const char odrediste[], const char oznaka_leta[], int kapija, int sat_polaska,
  int minute_polaska, int trajanje_leta);
void PostaviKasnjenje(int kasnjenje);
bool DaLiKasni() const;
int DajTrajanjeLeta() const;
std::pair<int, int> DajcekivanoVrijemePolijetanja() const;
std::pair<int, int> DajOcekivanoVrijemeSlijetanja() const;
void Ispisi() const;
```

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, uključujući i situaciju kada se za naziv odredišta ili za oznaku leta daju predugački nizovi znakova. Metoda "`PostaviKasnjenje`" postavlja informaciju o eventualnom kašnjenju na vrijednost zadanu parametrom, dok metoda "`DaLiKasni`" omogućava da se sazna da li odgovarajući let kasni ili ne (metoda vraća logičku vrijednost "**`true`**" u slučaju kašnjenja, a logičku vrijednost "**`false`**" u suprotnom slučaju). Metoda "`DajTrajanjeLeta`" daje kao rezultat trajanje leta u minutama, dok se pomoću metoda "`DajOcekivanoVrijemePolijetanja`" i "`DajOcekivanoVrijemeSlijetanja`" 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 "`Let`" na ekran. U slučaju da se radi o letu bez kašnjenja, ispis bi trebao da izgleda poput sljedećeg:

```
JFK 327   New York              12:50   19:30     5
```

Ovi podaci predstavljaju redom oznaku leta, naziv odredišta, vrijeme polijetanja, očekivano vrijeme slijetanja i broj izlazne kapije. Predvidite širinu od 10 mjesta za ispis šifre leta, 20 mjesta za naizv odredišta (ukoliko je naziv odredišta duži od 20 mjesta, treba ga sasijeći na tu dužinu), po 8 mjesta za vrijeme polijetanja i slijetanja, odnosno 6 mjesta za ispis broja izlazne kapije. Oznaka leta i naziv odredišta trebaju biti poravnati ulijevo u prostoru predviđenom za ispis, dok preostali podaci trebaju biti poravnati udesno. 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 letu koji kasni, ispis bi trebao da izgleda poput sljedećeg:

```
IST 932     Istanbul              15:50 (Planirano 15:30, Kasni 20 min)
```

Oznaku leta, naziv odredišta i vrijeme polijetanja formatirajte kao i u prethodnom slučaju, a dopunske informacije pišite u produžetku iza vremena polijetanja, na način koji je vidljiv iz prethodnog prikaza.

Klasa "`Letovi`" ima sljedeći interfejs:

```
explicit Letovi(int max_broj_letova);
Letovi(std::initializer_list<Let> lista_letova);
~Letovi();
Letovi(const Letovi &letovi);
Letovi(Letovi &&letovi);
Letovi &operator =(const Letovi &letovi);
Letovi &operator =(Letovi &&letovi);
void RegistrirajLet(const char odrediste[], const char oznaka_leta[], int kapija,
  int sat_polaska, int minute_polaska, int trajanje_leta);
void RegistrirajLet(Let *let);
int DajBrojLetova() const;
int DajBrojLetovaKojiKasne() const;
int DajProsjecnoTrajanjeLetova() const;
Let &DajPrviLet() const;
Let &DajPosljednjiLet() const;
void IsprazniKolekciju();
void Ispisi(int sati, int minute) const;
```

Podaci o letovima se čuvaju u dinamički alociranim objektima tipa "`Let`", 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 "`Letovi`" iz inicijalizacione liste čiji su elementi tipa "`Let`". 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 "`Letovi`" 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 "`RegistrirajLet`" podržana je u dvije verzije. Prva verzija kreira novi let u skladu sa parametrima (koji su identični kao kod konstruktora klase "`Let`") i registrira ga u kolekciji, dok druga verzija prosto kao parametar prihvata pokazivač na objekat tipa "`Let`" (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 letova" u slučaju da je dostignut maksimalan broj letova koji se mogu registrirati. Dalje, metode "`DajBrojLetova`", "`DajBrojLetovaKojiKasne`" i

"DajProsjecnoTrajanjeLetova" daju respektivno ukupan broj registriranih letova, broj letova koji kasne, te prosječno trajanje svih registriranih letova u minutama. Metodu "DajBrojLetovaKojiKasne" trebalo bi realizirati uz pomoć funkcije "count_if" iz biblioteke "algorithm" uz definiranje prikladne funkcije kriterija kao lambda funkcije. Metode "DajPrviLet" i "DajPosljednjiLet" daju kao rezultat prvi i posljednji let (tj. odgovarajući objekat tipa "Let") u toku dana (uključujući 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 letova, treba baciti izuzetak tipa "domain_error" uz prateći tekst "Nema registriranih letova". Metoda "IsprazniKolekciju" uklanja sve registrirane letove, 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 letovima, 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 letova vrši se prostim pozivom metode "Ispisi" nad objektima tipa "Let" 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.

22#. Izmijenite program iz prethodnog zadatka tako da se u klasi "Letovi" za evidenciju pokazivača na dinamički alocirane objekte tipa "Let" umjesto dinamički alociranog niza pokazivača koristiti vektor čiji su elementi pametni pokazivači na objekte tipa "Let", čime uklanjamo i ograničenje na maksimalno mogući broj polazaka koji se mogu registrirati, te rješavamo probleme vezane za oslobađanje memorije. Samim tim, konstruktor klase "Letovi" 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 "RegistrirajLet" zahtijevaće kao parametar pametni a ne obični pokazivač na objekat tipa "Let". 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.

23#. Za potrebe evidencije pacijenata koji dolaze na pregled kod ljekara, potrebno je implementirati klase "Datum", "Vrijeme", "Pregled" i "Pregledi". Klasa "Datum" je krajnje minimalistička klasa koja sadrži samo konstruktor, te metode "Postavi", "Ocitaj" i "Ispisi". Konstruktor i metoda "Postavi" omogućavaju inicijalizaciju i naknadnu izmjenu primjeraka ove klase, pri čemu se kao cjelobrojni parametri (tipa "int") zadaju redom dan, mjesec i godina. U slučaju da zadani podaci nisu legalni, baca se izuzetak tipa "domain_error" uz prateći tekst "Neispravan datum". Pristupna metoda "Ocitaj" treba da vrati pohranjeni dan, mjesec i godinu kao uređenu trojku (tj. objekat tipa "tuple") cijelih brojeva, dok metoda "Ispisi" treba da ispiše datum na ekran u obliku *dan*/*mjesec*/*godina* (npr. "25/5/2021").

Klasa "Vrijeme" je također minimalistička klasa i također sadrži samo konstruktor, te metode "Postavi", "Ocitaj" i "Ispisi" (kao i klasa "Datum"). Njen konstruktor i metoda "Postavi" očekuju dva cjelobrojna parametra (tipa "int") koji predstavljaju sate i minute u objektu tipa "Vrijeme" kojeg treba kreirati, odnosno postaviti. U slučaju da sati nisu u opsegu od 0 do 23 uključivo, a minute u opsegu od 0 do 59 uključivo, treba baciti izuzetak tipa "domain_error" uz prateći tekst "Neispravno vrijeme. Pristupna metoda "Ocitaj" treba da vrati pohranjene sate i minute kao uređeni par (tj. objekat tipa "pair") cijelih brojeva), dok metoda "Ispisi" treba da ispiše vrijeme na ekran u obliku *ss*:*mm* (npr. "09:54").

Klasa "Pregled" opisuje podatke o jednom zakazanom pregledu. Predviđeno je da ova klasa ima dva konstruktora. Prvi parametar u oba konstruktora je ime pacijenta (tipa "string"). Prvi konstruktor ima još dva parametra koji predstavljaju informacije o zakazanom danu i vremenu pregleda, i oni su tipa "Datum" i "Vrijeme" respektivno. Za razliku od ovog konstruktora, drugi konstruktor prima razbijene informacije o danu, mjesecu, godini, satu i minutama zakazanog pregleda kroz 5 cjelobrojnih parametara (tipa "int"). Dalje treba podržati funkcije članice nazvane "PromijeniPacijenta", "PromijeniDatum" i "PromijeniVrijeme", koje omogućavaju da se naknadno

promjene informacije o imenu pacijenta, odnosno o zakazanom datumu pregleda Njihovi parametri su novo ime pacijenta, novi datum (kao objekat tipa "`Datum`") odnosno novo vrijeme (kao objekat tipa "`Vrijeme`"). Sve ove tri funkcije kao rezultat treba da vrate referencu na izmijenjeni objekat tipa "`Pregled`" (sa ciljem da se omogući ulančano pozivanje). Pored ovih mutatorskih funkcija, treba podržati i mutatorske funkcije nazvane "`PomjeriDanUnaprijed`" i "`PomjeriDanUnazad`" koje pomjeraju zakazani datum pregleda za jedan dan unaprijed odnosno unazad. Ove funkcije nemaju parametara i ne vraćaju nikakav rezultat.

Od pristupnih metoda, treba podržati metode nazvane "`DajImePacijenta`", "`DajDatumPregleda`" i "`DajVrijemePregleda`", a koje vraćaju redom ime pacijenta, te datum i vrijeme zakazanog pregleda (sve ove metode su bez parametara). Predviđena je i statička funkcija članica "`DolaziPrije`" prima dva objekta tipa "`Pregled`" kao parametre i vraća "**`true`**" ukoliko je prvi pregled zakazan prije drugog, inače vraća "**`false`**". Konačno, predviđena je i funkcija članica "Ispisi" koja ispisuje podatke o pregledu na ekran na način da se prvo ispiše ime pacijenta, poravnato ulijevo u polju širine 30 znakova, a zatim datum i vrijeme pregleda koji su međusobno razdvojeni jednim razmakom (na kraju se prelazi u novi red).

Klasa "`Pregledi`" predstavlja kolekciju podataka o zakazanim pregledima. Podaci o pregledima se čuvaju u dinamički alociranim objektima, kojima se pristupa preko dinamički alociranog niza pokazivača (kojem se također pristupa putem odgovarajućeg pokazivača, koji je jedan od atributa klase). Konstruktor klase obavlja neophodnu alokaciju memorije, pri čemu njegov jedini parametar (tipa "**`int`**") predstavlja maksimalan broj pregleda koji se mogu registrirati. Pri tome, treba onemogućiti da se ovaj konstruktor koristi za automatsku konverziju cijelih brojeva u objekte tipa "`Pregledi`". Pored ovog konstruktora, treba podržati i sekvencijski konstruktor, koji omogućava kreiranje objekata tipa "`Pregledi`" iz inicijalizacione liste čiji su elementi tipa "`Pregled`", zatim destruktor koji oslobađa svu memoriju koja je zauzeta tokom života objekta, te kopirajući konstruktor i kopirajući operator dodjele, koji omogućavaju bezbjedno kopiranje i međusobno dodjeljivanje objekata tipa "`Pregledi`" korištenjem strategije dubokog kopiranja. Također treba podržati i pomjerajući konstruktor odnosno pomjerajući operator dodjele koji optimiziraju postupak kopiranja u slučajevima kada se kopiraju privremeni objekti tipa "`Pregled`", te koji omogućavaju podršku move-semantici za objekte tipa "`Pregled`".

Za registriranje pregleda, predviđene su tri verzije metode nazvane "`RegistrirajPregled`". Prve dvije verzije kreiraju novi pregled u skladu sa zadanim parametrima, koji su identični kao kod dva konstruktora klase "`Pregled`"), nakon čega ga registriraju u kolekciji, dok treća verzija prosto kao parametar prihvata pokazivač na neki objekat tipa "`Pregled`" (za koji pretpostavljamo da je već na neki način kreiran) i registira ga u kolekciji. Pri tome, u trećem slučaju, podrazumijeva se da klasa "`Pregledi`" preuzima vlasništvo nad registriranim objektom, tj. preuzima odgovornost za njegovo brisanje. U sva tri slučaja, treba baciti izuzetak tipa "`range_error`" uz prateći tekst "Dostignut maksimalni broj pregleda" u slučaju da je dostignut maksimalan broj pregleda koji se mogu registrirati.

Metode "`DajBrojPregleda`" i "`DajBrojPregledaNaDatum`" daju ukupan broj registriranih pregleda te broj pregleda zakazanih na zadani datum respektivno. Prva od njih nema parametara, dok druga ima parametar tipa "`Datum`" koji predstavlja željeni datum za koji se traži pregled. Pri tome, metodu "`DajBrojPregledaNaDatum`" obavezno treba realizirati uz pomoć funkcije "`count_if`" iz biblioteke "`algorithm`" uz definiranje prikladne funkcije kriterija kao lambda funkcije. Metoda "`DajNajranijiPregled`" bez parametara daje kao rezultat najraniji zakazani pregled (u vidu odgovarajućeg objekta tipa "`Pregled`"). Predviđene su dvije verzije ove metode, za konstantne i za nekonstantne objekte, pri čemu se za slučaj nekonstantnih objekata vraća referenca (da se izbjegne nepotrebno kopiranje i omogući izmjena vraćenog objekta). Ove metode treba realizirati putem funkcije "`min_element`" iz biblioteke "`algorithm`", uz odgovarajuću funkciju kriterija realiziranu kao lambda funkcija. U slučaju da nema registriranih pregleda, potrebno je baciti izuzetak tipa "`domain_error`" uz prateći tekst "Nema registriranih pregleda". Metoda "`IsprazniKolekciju`" bez parametara uklanja sve registrirane preglede, tako da nakon poziva ove metode kolekcija treba biti u identičnom stanju kakva je bila neposredno nakon kreiranja. Metoda "`ObrisiNajranijiPregled`" (također bez parametara) uklanja samo najraniji zakazani pregled (ova metoda se obično poziva nakon što se obradi odgovarajući pacijent). U slučaju da je kolekcija prazna, treba baciti izuzetak tipa "`range_error`" uz prateći tekst "Prazna kolekcija". Ni metoda "`ObrisiPregledePacijenta`" koja briše sve preglede pacijeta čije je ime zadano kao parametar (tipa "`string`"). Ukoliko nema registriranog ni

jednog pregleda sa zadanim imenom pacijenta, ne treba da se desi ništa. Sve ove metode ne vraćaju nikakav rezultat.

Za ispis informacija o pregledima, predviđena je metoda "`IspisiPregledeNaDatum`", koja ispisuje na ekran informacije o svim pregledima zakazanim datum koji joj se zadaje kao parametar, sortirane po vremenu u rastući poredak, dok metoda "`IspisiSvePreglede`" bez parametara ispisuje na ekran informacije o svim zakazanim pregledima, u hronološkom redoslijedu (tj. također sortirane po vremenu u rastući poredak). Obje metode ne vraćaju nikakav rezultat. Pri tome, ove metode treba da budu realizirane kao inspektori, odnosno da ne smiju promijeniti sadržaj objekta nad kojim se pozivaju, tako da se mogu primijeniti i nad konstantnim objektima. Ispis pojedinačnih pregleda vrši se prostim pozivom metode "`Ispisi`" nad objektima tipa "`Pregled`" pohranjenim u kolekciji.

Sve metode koje nisu trivijalne obavezno implementirajte izvan deklaracije klase. Također napišite i testni program u kojem ćete testirati sve elemente napisanih klasa (mogući izgled dijaloga biće specificiran putem javnih autotestova). Posebno se trebate uvjeriti da kopirajući i pomjerajući konstruktor te kopirajući i pomjerajući preklopljeni operator dodjele rade ispravno, kao i da ni u kom slučaju ne dolazi do curenja memorije.

24#. Izmijenite program iz prethodnog zadatka tako da se u klasi "`Pregledi`" za evidenciju pokazivača na dinamički alocirane objekte tipa "`Pregled`" umjesto dinamički alociranog niza pokazivača koristiti vektor čiji su elementi pametni pokazivači na objekte tipa "`Pregled`", čime uklanjamo i ograničenje na maksimalno mogući broj pregleda koji se mogu registrirati, te rješavamo probleme vezane za oslobađanje memorije. Samim tim, konstruktor klase "`Pregledi`" više neće imati parametar, dok metode za registraciju pregleda više ne trebaju provjeravati da li je dostignut maksimalan broj polazaka, s obzirom da ograničenje na maksimalan broj pregleda više ne postoji. Također, treća verzija funkcije "`RegistrirajPregled`" zahtijevaće kao parametar pametni a ne obični pokazivač na objekat tipa "`Pregled`". 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.

25#. Implementirajte jednostavan program koji olakšava vođenje administrativnih poslova u nekoj videoteci. Program se zasniva na tri klase "`Korisnik`", "`Film`" i "`Videoteka`". Primjerci ovih klasa modeliraju respektivno korisnike videoteke, filmove u videoteci te samu videoteku (ova posljednja klasa je tzv. singlton klasa, što znači da će u čitavom programu biti samo jedan primjerak te klase).

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 4 parametara koji inicijalizira sve atribute na vrijednosti zadane parametrima (redoslijed parametara je isti kao i redoslijed gore navedenih atributa), zatim 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 na korisnika koji je zadužio film odnosno 0 ukoliko film nije zadužena, dok su ostali atributi stringovnog tipa. Interfejs klase sadrži konstruktor sa 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 "`DajKodKogaJe`" koje vraćaju vrijednosti odgovarajućih atributa, metodu "`DaLiJeDVD`" koja vraća informaciju da li je film na DVD-u ili ne, te metode "`ZaduziFilm`", "`RazduziFilm`", "`DaLiJeZaduzen`" 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, dok 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 varijablama, kojima se pristupa pomoću dva vektora čiji su elementi pokazivači na korisnike (tj. na objekte tipa "Korisnik") odnosno pokazivači na fimove (tj. na objekte tipa "Film"). Ova dva vektora su ujedno i jedini atributi klase "Videoteka". Broj elemenata ovih vektora nije unaprijed određen, nego raste po potrebi sa dodavanjem novih korisnika odnosno filmova u evidenciju. Klasi nije potreban ručno pisani konstruktor (s obzirom da će atributi koji su vektori svakako biti automatski inicijalizirani na prazne vektore), ali treba imati destruktor koji će po završetku postojanja objekta tipa "Videoteka" obrisati sve korisnike i filmove koji su evidentirani u njoj (tj. koji su u njenom vlasništvu). Kopiranje i međusobno dodjeljivanje objekata tipa "Videoteka" treba zabraniti. Pored ovih elemenata, klasa "Videoteka" sadrži i nekoliko metoda. 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. U slučaju da već postoji korisnik sa istim članskim brojem, metoda baca izuzetak. Metoda "RegistrirajNoviFilm" radi analognu stvar, ali za filmove (tj. objekte tipa "Film"). Metoda "NadjiKorisnika" prima kao parametar članski broj korisnika i vraća kao rezultat referencu na korisnika sa zadanim članskim brojem, ili baca izuzetak ako takvog korisnika nema. Metoda "NadjiFilm" radi analognu stvar, ali za filmove (parametar je evidencijski broj). Metoda "IzlistajKorisnike" ispisuje podatke o svim registriranim korisnicima, jedan za drugim, sa po jednim praznim redom između svaka dva korisnika, dok metoda "IzlistajFilmove" tu istu stvar radi za filmove. Pri tome, ukoliko je film zadužen, iza standardnih podataka koji se ispisuju za film treba ispisati i tekst "Zadužen kod korisnika: " iza čega slijedi korisnički broj korisnika koji je zadužio film. 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. 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 zadužen, metoda baca izuzetak. 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 "IzlistajKorisnike"). U slučaju da korisnik nije zadužio niti jedan film, metoda ispisuje tekst "Korisnik nema zaduženja!", dok u slučaju da je članski broj neispravan, metoda baca izuzetak.

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.

26#. Elektrotehnički fakultet u Sarajevu uveo je mogućnost iznajmljivanja laptopa studentima, odnosno svaki student po veoma povoljnoj cijeni može iznajmiti laptop za potrebe studija. Da bi se olakšalo vođenje evidencije o zaduženjima, potrebno je napraviti program koji se zasniva na tri klase "Student", "Laptop" i "Administracija". Primjerci prve dvije klase modeliraju respektivno studente i laptope, dok klasa "Administracija" predstavlja bazu podataka koja u sebi čuva evidenciju o svim registriranim studentima, laptopima i realiziranim zaduženjima.

Klasa "Student" sadrži konstruktor sa 5 parametara koji predstavljaju redom broj indeksa studenta (cijeli broj), godinu studija (objašnjenje slijedi kasnije), ime i prezime studenta, adresu studenta, te broj telefona studenta (posljednja četiri parametra su tipa "string"). Broj indeksa studenta mora biti peterocifreni broj. Kao godina studija, može se zadati jedan od sljedećih nizova znakova: "1", "2", "3", "1/B", "2/B", "3/B", "1/M", "2/M", "1/D", "2/D" ili "3/D". "B" odgovara bachelor studiju, "M"

master studiju, a "D" doktorskom studiju, dok su "1", "2" i "3" sinonimi za "1/B", "2/B" i "3/B" respektivno. Bilo koji drugi niz znakova smatra se ilegalnim. Ime i prezime i adresa mogu biti bilo kakvi, ali se svi suvišni razmaci (tj. razmaci na početku ili kraju, kao i višestruki razmaci između riječi) ignoriraju, tako da ukoliko se, na primjer, kao ime zada " Huso Husic Car ", to treba biti pohranjeno kao "Huso Husic Car". Broj telefona mora biti u formatu *cifre*/*cifre*−*cifre* (npr. 037/427−3421). Pri tome, *nema ograničenja na broj uzastopnih cifara* (tako da se i broj 1/1−1 također smatra korektnim). Ukoliko bilo koji od parametara nije ispravan, treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Neispravni parametri". Pored konstruktora, klasa "`Student`" sadrži još i pristupne metode "`DajIndeks`", "`DajGodinuStudija`", "`DajImePrezime`", "`DajAdresu`" i "`DajTelefon`" bez parametara kojima se mogu pročitati informacije o studentu (posljednje četiri metode vraćaju rezultat tipa "`string`"), te metodu "`Ispisi`" bez parametara koja ispisuje podatke o studentu na ekran. Metoda "`DajGodinuStudija`" treba uvijek da vrati punu varijantu zapisa o godini studija, odnosno ukoliko je godina studija registrirana kao "1", metoda treba da vrati "1/B" a ne "1". Format ispisa treba da bude kao u sljedećem primjeru (iza dvotačke je tačno jedan razmak):

```
Broj indeksa: 35124
Godina studija: 3/B
Ime i prezime: Huso Husic Car
Adresa: Trg zrtava reforme obrazovanja 25
Telefon: 069/434-072
```

Klasa "`Laptop`" treba posjedovati četiri atributa nazvana "`ev_broj`", "`naziv`", "`karakteristike`" i "`kod_koga_je`" (imena su bitna, ovo će se testirati) koji redom predstavljaju evidencijski broj laptopa (cijeli broj), naziv laptopa (tipa "`string`"), napomene o osnovnim karakteristikama laptopa (tipa "`string`"), te pokazivač na studenta koji je eventualno zadužio laptop, ili "**`nullptr`**"ukoliko laptop nije zadužen. Konstruktor klase ima tri parametra i vrši inicijaliziju evidencijskog broja, naziva i napomena o karakteristikama laptopa na vrijednosti zadane parametrima (drugi i treći parametar su tipa "`string`"), te evidentira da laptop trenutno nije zadužen. Prvi parametar mora biti nenegativan, u suprotnom se baca izuzetak tipa "`domain_error`" uz prateći tekst "Neispravni parametri". Pored konstruktora, interfejs klase sadrži pristupne metode "`DajEvidencijskiBroj`", "`DajNaziv`" i "`DajKarakteristike`" (bez parametara) koje redom vraćaju evidencijski broj, naziv i napomene o karakteristikama laptopa (posljednje dvije metode vraćaju rezultat tipa "`string`"), kao i metode "`Zaduzi`", "`Razduzi`", "`DaLiJeZaduzen`", "`DajKodKogaJe`", "`DajPokKodKogaJe`" i "`Ispisi`". Metoda "`Zaduzi`" vrši zaduživanje laptopa, a parametar joj je referenca na studenta koji zadužuje laptop. Metoda treba da baci izuzetak tipa "`domain_error`" uz prateći tekst "Laptop vec zaduzen" ukoliko je laptop već zadužen. Metoda "`Razduzi`" nema parametara, a vrši razduživanje laptopa. Ova metoda ne radi ništa ukoliko laptop uopće nije zadužen. Metoda "`DaLiJeZaduzen`" također nema parametara i prosto vraća logičku informaciju da li je laptop zadužen ili ne, dok metoda "`DajKodKogaJe`" (isto bez parametara) daje kao rezultat referencu na studenta koji je zadužio laptop (ili baca izuzetak tipa "`domain_error`" uz prateći tekst "Laptop nije zaduzen" ukoliko laptop nije zadužen). Slična je i metoda "`DajPokKodKogaJe`", samo što ona nikad ne baca izuzetak, nego vraća kao rezultat pokazivač na studenta koi je zadužio laptop, ili "**`nullptr`**" ako laptop nije zadužen. Konačno, metoda "`Ispisi`" (bez parametara) vrši ispis podataka o laptopu na ekran, na način kao u sljedećem primjeru:

```
Evidencijski broj: 724
Naziv: ASUS X554L
Karakteristike: Intel CPU 2.4 GHz, 8 GB RAM
```

Klasa "`Administracija`" objedinjuje u sebi podatke o svim studentima, kao i o svim raspoloživim laptopima. Radi brže pretrage, podaci se čuvaju u dvije mape, mapa studenata i mapa laptopa, i one su jedini atributi ove klase. Svaki student i svaki laptop imaju jedinstveni identifikacioni broj (broj indeksa odnosno evidencijski broj laptopa), i oni su ključna polja ove dvije mape. Ostatak podataka o studentima odnosno laptopima nalazi se u dinamički alocianim objektima tipa "`Student`" odnosno "`Laptop`", tako da su pridružene vrijednosti u mapi knjiga odnosno mapi korisnika (obični) pokazivači na dinamički alocirane objekte koje sadrže podatke o odgovarajućem studentu odnosno laptopu. Objekti tipa "`Administracija`" 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 "`Administracija`" treba podržavati i odgovarajuće metode za manipulaciju sa podacima. Metoda "`RegistrirajNovogStudenta`" prima kao parametre podatke o novom studentu (ovi parametri su isti kao parametri konstruktora klase "`Student`"), nakon čega se kreira

odgovarajući objekat tipa "Student" i upisuje u evidenciju. U slučaju da već postoji student sa istim brojem indeksa, metoda baca izuzetak tipa "domain_error" uz prateći tekst "Student s tim indeksom vec postoji". Sličnu stvar radi i metoda "RegistrirajNoviLaptop", ali za laptope (tj. objekte tipa "Laptop"), pri čemu je prateći tekst uz izuzetak "Laptop s tim evidencijskim brojem vec postoji". Metoda "NadjiStudenta" prima kao parametar broj indeksa i vraća kao rezultat referencu na studenta sa zadanim brojem indeksa, ili baca izuzetak tipa "domain_error" uz prateći tekst "Student nije nadjen" ako takvog studenta nema. Ukoliko se ova metoda pozove nad konstatntim objektom tipa "Administracija", umjesto reference na studenta treba da se vrati njegova kopija (tj, novi objekat tipa "Student" istovjetan nađenom). Metoda "NadjiLaptop" radi analognu stvar, ali za laptope (parametar je evidencijski broj, a prateći tekst uz izuzetak je "Laptop nije nadjen"). Metoda "IzlistajStudente" ispisuje podatke o svim registriranim studentima, jedan za drugim, sa po jednim praznim redom između svaka dva studenta, dok metoda "IzlistajLaptope" tu istu stvar radi za laptope. Pri tome, ukoliko je laptop zadužen, iza standardnih podataka koji se ispisuju za laptop treba odmah ispod u novom redu ispisati i tekst "Zaduzio(la): " iza čega slijedi ime i prezime studenta, te njegov broj indeksa u zagradi (recimo "Huso Husic Car (35124)"). Metoda "ZaduziLaptop" prima kao parametre broj indeksa studenta koji želi zadužiti laptop, te evidencijski broj laptopa koji se zadužuje. Ona vrši registraciju da je navedeni laptop zadužen kod navedenog studenta. U slučaju da se ne može naći student sa zadanim indeksom ili laptop sa zadanim evidencijskim brojem, metoda baca izuzetak tipa "domain_error" uz prateće tekstove "Student nije nadjen" odnosno "Laptop nije nadjen" (prvo se testira student pa laptop, tako da ukoliko nije nađen ni student ni laptop, prijavljuje se prvi tekst). Ukoliko je laptop već zadužen, također treba baciti izuzetak istog tipa, ali uz prateći tekst "Laptop vec zaduzen". Izuzetak (istog tipa) uz prateći tekst "Student je vec zaduzio laptop" treba baciti i u slučaju da je taj student već zadužio neki laptop. Metoda "NadjiSlobodniLaptop" bez parametara pronalazi prvi registrirani slobodni laptop (tj. koji nije zadužen ni kod koga) i vraća njegov evidencijski broj kao parametar, ili baca izuzetak tipa "domain_error" uz prateći tekst "Nema slobodnih laptopa" ukoliko takav ne postoji. Metoda "RazduziLaptop" prima kao parametar evidencijski broj laptopa. Ona registrira da laptop više nije zadužen. U slučaju da takav laptop nije nađen, ili ukoliko on uopće nije zadužen, metoda baca izuzetak tipa "domain_error" uz prateće tekstove "Laptop nije nadjen" odnosno "Laptop nije zaduzen". Konačno, metoda "PrikaziZaduzenja" bez parametara ispisuje podatke o svim zaduženjima u vidu rečenica oblika "Student Huso Husic Car (35124) zaduzio/la laptop broj 724" (svaka rečenica u posebnom redu) ili tekst "Nema zaduzenja" ukoliko niti jedan student nije zadužio niti jedan laptop.

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 "Administracija". 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.

27#. Neki studenti su pitali predmetnog nastavnika i druge članove nastavnog ansambla zašto ih maltretiramo sa ručnim upravljanjem alokacijom memorije i drugim 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 "Administracija" i dalje trebaju kopirati kao duboke kopije, jer ovo je C++ a ne Java (imaćete prilike u Javi uživati u čarima plitkih kopija).

28#. Za potrebe evidencije prodaje stanova nekog stambenog fonda, potrebno je implementirati klase "Datum", "Kupac", "Stan", "Prodaja" i "Prodaje". Klasa "Datum" je minimalistička klasa sa sljedećim interfejsom:

```
Datum(int dan, int mjesec, int godina);
void Postavi(int dan, int mjesec, int mjesec);
int DajDan() const;
int DajMjesec() const;
int DajGodinu() const;
void Ispisi() const;
```

Klasa "Kupac" je također minimalistička, sa sljedećim interfejsom:

```
Kupac(const std::string &ime_i_prezime, const Datum &datum_rodjenja);
void Postavi(const std::string &ime_i_prezime, const Datum &datum_rodjenja);
std::string DajImePrezime() const;
Datum DajDatumRodjenja() const;
void Ispisi() const;
```

Konstruktor i metoda "Postavi" za obje ove klase omogućava inicijalizaciju i naknadnu izmjenu primjeraka ove klase, pri čemu se baca izuzetak tipa "domain_error" uz prateći tekst "Neispravan datum" odnosno "Neispravno ime i prezime" u slučaju da zadani podaci nisu legalni. Ime i prezime se smatraju nelegalnim ako sadrže bilo koji znak koji nije slovo, broj ili znak "crtica", razmak ili apostrof. Trivijalne pristupne metode "DajDan", "DajMjesec", "DajGodinu", "DajImePrezime" i "DajDatumRodjenja" samo vraćaju vrijednosti odgovarajućih atributa. Metoda "Ispisi" za klasu "Datum" ispisuje datum na ekran u obliku *dan/mjesec/godina* (npr. "23/5/2016"), dok istoimena metoda za osobu treba ispisati ime i prezime osobe i njen datum rođenja u obliku: *ime_i_prezime* (*dan/mjesec/godina*), npr. "Niko Nikic (17/5/1986)".

Klasa "Stan" ima sljedeći interfejs:

```
Stan(const std::string &adresa, int sprat, int broj_soba, bool namjesten,
  double kvadratura);
void Postavi(const std::string &adresa, int sprat, int broj_soba,
  bool namjesten, double kvadratura);
std::string DajAdresu() const;
int DajSprat() const;
int DajBrojSoba() const;
bool DajNamjesten() const;
double DajKvadraturu() const;
void Ispisi() const;
```

Konstruktor i funkcija "Postavi" postavljaju atribute klase na zadane vrijednosti. Ukoliko neki od parametara ima negativnu vrijednost, baca se izuzetak tipa "domain_error" uz prateći tekst "Neispravan unos podataka". Trivijalne pristupne metode samo vraćaju vrijednosti odgovarajućih atributa. Metoda "Ispisi" za klasu "Stan" ispisuje na ekran vrijednosti njenih atributa formatu "Stan se nalazi na adresi *adresa* na *sprat* spratu i ima *broj_soba* soba. Kvadratura stana je *kvadratura* i stan *je_ili_nije* namjesten". Na primjer:

```
Stan se nalazi na adresi Hamdije Cemerlica 14 na 5. spratu i ima 5 soba. Kvadratura stana
je 35.45 (m^2) i stan je namjesten.

Stan se nalazi na adresi Hamdije Cemerlica 14 na 1. spratu i ima 2 sobe. Kvadratura stana
je 39.21 (m^2) i stan nije namjesten.
```

Vodite računa o gramatičkoj korektnosti ispisa. Konkretno, treba voditi računa da li se broj sobe završava na 1, 2, 3 ili 4, tako da npr. za 3 sobe treba pisati "… i ima 3 sobe …" a ne "… i ima 3 soba…".

Klasa "Prodaja" opisuje podatke o jednoj planiranoj prodaji, a ima sljedeći interfejs:

```
Prodaja(const std::string &ime_agenta_prodaje, double cijena_stana,
  const Datum &datum_prodaje, const Kupac &kupac_stana,
  const Stan &kupljeni_stan);
Prodaja(const std::string &ime_agenta_prodaje, double cijena_stana,
  int dan_prodaje, int mjesec_prodaje, int godina_prodaje,
  std::string &ime_kupca, const Datum &datum_rodjenja_kupca,
  const std::string &adresa_stana, int sprat_stana, int broj_soba,
  bool namjesten_stan, double broj_kvadrata);
void PromijeniKupca(const Kupac &novi_kupac);
void PromijeniStan(const Stan &novi_stan);
void PromijeniDatumKupovine(const Datum &novi_datum);
void PromijeniCijenuProdaje(const double &nova_cijena);
void PomjeriDanUnaprijed();
void PomjeriDanUnazad();
std::string DajImeAgenta() const;
std::string DajImeKupca() const;
Datum DajDatumProdaje() const;
double DajCijenuStana() const;
friend bool ProdatPrije(const Prodaja &p1, const Prodaja &p2);
friend bool SkupljiStan(const Prodaja &p1, const Prodaja &p2);
void Ispisi() const;
```

Konstruktori omogućavaju kreiranje jednog objekta tipa "Prodaja". U oba slučaja, ime agenta prodaje je prvi parametar, a cijena stana drugi. Podržana su dva konstruktora, jedan prima informacije o zakazanom datumu prodaje, kupcu stana i stanu o kojem je riječ putem objekata tipa "Datum", "Kupac" i "Stan" respektivno, dok drugi prima razbijene informacije o danu, mjesecu, godini prodaje, imenu i prezimenu kupca te njegovom datumu rođenja i naposlijetku o osobinama kupljenog stana.

Uz pomoć funkcija "PromijeniKupca", "PromijeniDatumKupovine", "PromijeniCijenuKupovine" i "PromijeniStan" mogu se naknadno promjeniti informacije o imenu kupca, odnosno zakazanom datumu prodaje, cijeni stana, ili stanu koji kupac namjerava kupiti. Pored toga, treba podržati i funkcije "PomjeriDanUnaprijed" i "PomjeriDanUnazad" koje pomjeraju zakazani datum prodaje za jedan dan unaprijed odnosno unazad.
Pristupne metode "DajImeAgenta", "DajImeKupca", "DajDatumProdaje" i "DajCijenuStana" vraćaju redom ime agenta prodaje, ime kupca stana, te datum i i cijenu stana koji se prodaje. Prijateljska funkcija "ProdatPrije" prima dva objekta tipa "Prodaja" kao parametre i vraća "**true**" ukoliko je prva prodaja zakazana prije druge, inače vraća "**false**". Konačno, funkcija "Ispisi" ispisuje podatke o prodaji na sljedeći način:

```
Ime agenta:                     Niko Nikic
Ime kupca:                      Mujo Mujic (21/9/1982)
Zakazani datum prodaje:         27/5/2017
Cijena stana:                   78985
Informacije o stanu:
Stan se nalazi na adresi Hamdije Cemerlica 14 na 5. spratu i ima 5 soba. Kvadratura stana
je 35.45 (m^2) i stan je namjesten.
```

Klasa "Prodaje" predstavlja kolekciju podataka o zakazanim prodajama, izvedenu kao dinamički alocirani niz pokazivača na objekte tipa "Prodaja". Ova klasa sadrži sljedeći interfejs:

```
explicit Prodaje(int max_broj_prodaja);
Prodaje(std::initializer_list<Prodaja> spisak_prodaja);
~Prodaje();
Prodaje(const Prodaje &prodaje);
Prodaje(Prodaje &&prodaje);
Prodaje &operator =(const Prodaje &prodaje);
Prodaje &operator =(Prodaje &&prodaje);
void RegistrirajProdaju(const std::string &ime_agenta_prodaje,
  double cijena_stana, const Datum &datum_prodaje, const Kupac &kupac_stana,
  const Stn &kupljeni_stan);
void RegistrirajProdaju(const std::string &ime_agenta_prodaje, int dan_prodaje,
  int mjesec_prodaje, int godina_prodaje, std::string &ime_kupca,
  const Datum &datum_rodjenja_kupca, const std::string &adresa_stana,
  int sprat_stana, int broj_soba, bool namjesten_stan, double broj_kvadrata);
void RegistrirajProdaju(Prodaja *prodaja);
int DajBrojProdaja() const;
int DajBrojProdajaNaDatum(const Datum &datum) const;
int DajBrojProdajaOdAgenta(const std::string &ime_agenta) const;
Prodaja &DajNajranijuProdaju();
Prodaja DajNajranijuProdaju() const;
Prodaja &DajNajskupljuProdaju();
Prodaja DajNajskupljuProdaju() const;
void IsprazniKolekciju();
void ObrisiNajranijuProdaju();
void ObrisiProdajeAgenta(const std::string &ime_agenta);
void ObrisiProdajeNaDatum(const Datum &datum);
void IspisiProdajeNaDatum(const Datum &datum) const;
void IspisiSveProdaje() const;
```

Konstruktor obavlja neophodnu alokaciju memorije, pri čemu parametar konstruktora predstavlja maksimalan broj prodaja koji se mogu registrirati. Predviđen je i sekvencijski konstruktor, koji omogućava kreiranje objekata tipa "Prodaje" iz inicijalizacione liste čiji su elementi tipa "Prodaja". 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 "Prodaje" korištenjem strategije dubokog kopiranja. Također su predviđeni i pomjerajući konstruktor odnosno pomjerajući operator dodjele koji optimiziraju postupak kopiranja u slučajevima kada se kopiraju privremeni objekti.

Metoda "RegistrirajProdaju" podržana je u tri verzije. Prve dvije verzije kreiraju novu prodaju u skladu sa parametrima (koji su identični kao kod konstruktora klase "Prodaja") i registriraju je u kolekciji, dok treća verzija prosto kao parametar prihvata pokazivač na objekat tipa "Prodaja" (za koji pretpostavljamo da je već na neki način kreiran) i registira ga u kolekciji. Registracijom se objekti tipa "Prodaja" predaju u vlasništvo objektu tipa "Prodaje", tako da je on odgovoran i za njihovo kasnije brisanje. U sva tri slučaja, treba baciti izuzetak tipa "range_error" uz prateći tekst "Dostignut maksimalni broj prodaja" u slučaju da je dostignut maksimalan broj prodaja koje se mogu registrirati.

Metode "DajBrojProdaja" i "DajBrojProdajaNaDatum" daju ukupan broj registriranih prodaja te broj prodaja zakazanih na zadani datum respektivno. Metodu "DajBrojProdajaNaDatum" trebalo bi realizirati uz pomoć funkcije "count_if" iz biblioteke "algorithm" uz definiranje prikladne funkcije kriterija kao lambda funkcije. Metoda "DajNajranijuProdaju" daje kao rezultat najranije zakazanu prodaju (tj. odgovarajući objekat tipa "Prodaja"). Treba podržati dvije verzije ove metode, pri čemu se za slučaj nekonstantnih objekata vraća referenca (da se izbjegne nepotrebno kopiranje i omogući izmjena vraćenog objekta). Isto vrijedi i za metodu "DajNajskupljuProdaju" koja kao rezultat daje najskuplju prodaju koja je zakazana (u odnosu na cijenu stana). Ove metode treba realizirati putem funkcije "min_element" i "max_element" iz biblioteke "algorithm", uz odgovarajuću funkciju kriterija realiziranu kao lambda funkcija. U slučaju da nema registriranih prodaja, obje funkcije trebaju baciti izuzetak tipa "domain_error" uz prateći tekst "Nema registriranih prodaja".

Metoda "IsprazniKolekciju" uklanja sve registrirane prodaje, tako da nakon poziva ove metode kolekcija treba biti u identičnom stanju kakva je bila neposredno nakon kreiranja. Metoda "ObrisiNajranijuProdaju" uklanja samo najraniju zakazanu prodaju (ova metoda se obično poziva nakon što izvrši zaključivanje kupovine). U slučaju da je kolekcija prazna, treba baciti izuzetak tipa "range_error" uz prateći tekst "Prazna kolekcija". Metoda "ObrisiProdajeAgenta" briše sve zakazane prodaje agenta čije je ime i prezime zadano kao parametar. Ukoliko nema registrirane nijedne prodaje sa zadanim imenom agenta, ne treba da se desi ništa. Metoda "ObrisiProdajeNaDatum" briše sve zakazane prodaje na datum koji joj je zadan kao parametar. Ukoliko nema registrirane nijedne prodajena dati datum, ne treba da se desi ništa.

Metoda "IspisiProdajeNaDatum" ispisuje informacije o svim prodajama zakazanim na zadani datum, sortirane u rastući poredak po imenu kupaca, dok metoda "IspisiSveProdaje" ispisuje informacije o svim zakazanim prodajama, u hronološkom redoslijedu (tj. sortirane po datumu u rastući poredak). U slučaju da ima više prodaja zakazanih na isti datum, takve prodaje između sebe trebaju biti sortirane po imenu kupaca. Pri tome, vodite računa da su ove metode realizirane kao inspektori, odnosno da ne smiju promijeniti sadržaj objekta nad kojim se pozivaju. Ispis pojedinačnih prodaja vrši se prostim pozivom metode "Ispisi" nad objektima tipa "Prodaja" pohranjenim u kolekciji.

Sve metode obavezno implementirajte izvan deklaracije klase, osim trivijalnih metoda koje možete implementirati direktno unutar deklaracije klase. Potrebno je napisati i testni program u kojem će korisnik putem menija birati koje transakcije želi da obavlja. Naravno, svakako je potrebno prvo da istestirate funkcionalnost svih elemenata napisanih klasa. Posebno se trebate uvjeriti da kopirajući i pomjerajući konstruktor, te kopirajući i pomjerajući operator dodjele rade ispravno, kao da ni u kom slučaju ne dolazi do curenja memorije.

29#. Izmijenite program iz prethodnog zadatka tako da se u klasi "Prodaje" za evidenciju pokazivača na dinamički alocirane objekte tipa "Prodaja" umjesto dinamički alociranog niza pokazivača koristiti vektor čiji su elementi pametni pokazivači na objekte tipa "Prodaja", čime uklanjamo i ograničenje na maksimalno mogući broj prodaja koje se mogu registrirati, te rješavamo probleme vezane za oslobađanje memorije. Samim tim, konstruktor klase "Prodaje" više neće imati parametar, dok metode za registraciju prodaja više ne trebaju provjeravati da li je dostignut maksimalan broj prodaja, s obzirom da ograničenje na maksimalan broj prodaja više ne postoji. Također, treća verzija funkcije "RegistrirajProdaju" zahtijevaće kao parametar pametni a ne obični pokazivač na objekat tipa "Prodaja". 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.

30*. Napišite generičku klasu "`Skup`" koja oponaša najosnovnije funkcionalnosti bibliotečnog tipa "`set`" putem svog jedinog privatnog atributa koji je tipa "`vector`". Potrebno je podržati konstruktor bez parametara koji kreira prazan "`Skup`", te sekvencijalni konstruktor koji prihvata inicijalizacionu listu i na osnovu nje popunjava atribut-vektor, uz izostavljanje duplikata. Od metoda treba podržati metodu "`Velicina`" koja vraća broj elemenata "`Skup`"-a, zatim "`Dodaj`" koja u "`Skup`" dodaje novi element proslijeđen kao parametar, vodeći računa da elementi ostanu sortirani, te metodu "`Izbrisi`" koja uklanja element proslijeđen kao parametar (vektor treba ostati sortiran). U slučaju da element već postoji u "`Skup`"-u, metoda "`Dodaj`" ne treba uraditi ništa, što je u skladu sa ponašanjem metode "`insert`" tipa "`set`". Isto vrijedi i za metodu "`Izbrisi`" u slučaju da element nije pronađen. Konačno, treba implementirati i metodu "`ImaLiGa`" koja testira da li se element zadan kao parametar nalazi u "`Skup`"-u ili ne (ona vraća odgovarajuću logičku vrijednost kao rezultat), te metodu "`Ispisi`" koja ispisuje sve elemente "`Skup`"-a na ekran (naravno, u sortiranom poretku).