Zadaća 4

Ova zadaća nosi ukupno 6 poena. Zadatak 1, 3 i 5 nose po 1.2 poena, zadatak 2 nosi 1 poen, dok zadaci 4 i 6 zadatak nose po 0.7 poena (zadaci 4 i 6 su mala prepravka zadataka 3 i 5 i smiju se kopirati). Svi zadaci se mogu uraditi na osnovu gradiva sa prvih jedanaest predavanja i pretpostavljenog predznanja iz predmeta "Osnove računarstva". Rok za predaju ove zadaće je nedjelja, 9. VI 2024. do kraja dana.

NAPOMENA: U slučaju da smatrate da Vam u zadacima trebaju neke pomoćne funkcije za realizaciju neophodnih funkcionalnosti (a trebale bi Vam, ako ne želite programe sa više od 500 linija koda), možete ih slobodno dodati u privatni dio klase. U svim zadacima, *sve metode koje po prirodi svog zadatka trebaju biti inspektori*, *treba obavezno deklarirati kao inspektore*.

1. 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 "$\{\{x', y'\},\{x', y'\}\}$" gdje 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). Tačan izgled dijaloga između korisnika i programa biće specificiran putem javnih autotestova.

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).

2. 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.

Očigledno, 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. 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 građane). Međutim, postoji rješenje koje je mnogo bolje sa aspekta utroška resursa (koje trebate izvesti u ovoj zadaći). Svaki objekat tipa "`GradjaninBiH`" će u sebi sadržavati jedan pokazivač na posljednjeg građanina koji je kreiran prije njega (osim prvog kreiranog građanina koji će na tom mjestu imati nul-pokazivač), tako da će svi kreirani objekti tipa "`GradjaninBiH`" 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

posljednjeg kreiranog građanina (ili nul-pokazivač ukoliko niti jedan građanin nije kreiran). Ovaj atribut je potreban da bismo znali gdje počinje "lanac" povezanih građana. Sad se test na jednakost JMBG izvodi tako što se prođe kroz čitavu listu i testira da li je JMBG građanina kojeg želimo kreirati jednak JMBG ma kojeg elementa liste. Ukoliko testiranje prođe uspješno, novokreirani građanin se također "uvezuje" u listu. Interesantno je da će klasa "`GradjaninBiH`" 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. Na sličan način se izvodi i postupak pretrage koji su slobodni kôdovi u drugoj varijanti konstruktora koja treba automatski generirati JMBG.

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. Nepoštovanje ove zabrane biće kažnjeno davanjem 0 poena na čitav zadatak!

3. 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.

4. 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.

5. 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.

6. 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).