Zadaća 5.

Ova zadaća nosi ukupno 5 poena. Prvi zadatak nosi 1,2 poena, drugi 0,4 poena, treći i četvrti zadatak po 1,2 poena i peti zadaak 1 poen. Svi zadaci se mogu uraditi na osnovu gradiva sa prvih dvanaest predavanja i pretpostavljenog predznanja iz predmeta "Osnove računarstva". Rok za predaju ove zadaće je subota, 17. VI 2017. do 9.00.

NAPOMENA: U slučaju da smatrate da Vam u zadacima trebaju neke pomoćne funkcije za realizaciju neophodnih funkcionalnosti (što je preporučljivo, da rješenja ne budu predugačka), možete ih slobodno dodati u privatni dio klase. Međutim, interfejs klase *ne smijete mijenjati*, osim ukoliko se postavke zadatka jasno ne vidi da treba praviti izmjene u interfejsu klase.

1. Implementirajte jednostavan program koji olakšava vođenje administrativnih poslova u nekoj biblioteci. Korisnike (klijente) biblioteke modelira struktura (ne klasa) "`Korisnik`", koja sadrži samo četiri atributa tipa "`string`", nazvana redom "`ime`", "`prezime`", "`adresa`" i "`telefon`", koji redom sadrže podatke o imenu, prezimenu, adresi i broju telefona korisnika. Knjige u biblioteci modelira klasa "`Knjiga`" čiji privatni atributi čuvaju informacije o naslovu knjige, imenu pisca, žanru i godini izdavanja, kao i informaciju o eventualnom zaduženju knjige. Naslov, ime pisca i žanr su tipa "`string`", godina izdavanja je cijeli broj, dok je informacija o zaduženju pokazivač na korisnika koji je zadužio knjigu odnosno "**`nullptr`**" ukoliko knjiga nije zadužena. Interfejs klase sadrži konstruktor sa 4 parametra koji inicijalizira sve atribute klase na vrijednosti zadane parametrima (redoslijed parametara je isti kao i redoslijed gore navedenih atributa), osim informacije o zaduženju koja se postavlja tako da signalizira da knjiga nije zadužena. Pored konstruktora, interfejs klase sadrži pristupne metode "`DajNaslov`", "`DajAutora`", "`DajZanr`", "`DajGodinuIzdavanja`" i "`DajKodKogaJe`" koje prosto vraćaju vrijednosti odgovarajućih atributa, te metode "`ZaduziKnjigu`", "`RazduziKnjigu`" i "`DaLiJeZaduzena`". Metoda "`ZaduziKnjigu`" vrši zaduživanje knjige, a parametar joj je referenca na korisnika koji zadužuje knjigu. Metoda "`RazduziKnjigu`" nema parametara, a vrši razduživanje knjige. Konačno, metoda "`DaLiJeZaduzena`" također nema parametara i prosto vraća logičku informaciju da li je knjiga zadužena ili ne.

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

Clanski broj: <***članski_broj***>
Ime i prezime: <***ime***> <***prezime***>
Adresa: <***adresa***>
Broj telefona: <***broj_telefona***>

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

Pri tome, ukoliko je knjiga zadužena, treba još dodatno odmah ispod u novom redu ispisati i tekst "Zaduzena kod korisnika: " iza čega slijedi ime i prezime (sa jednim razmakom između) korisnika biblioteke koji je zadužio knjigu. Metoda "`ZaduziKnjigu`" prima kao parametre evidencijski broj knjige, kao i članski broj korisnika koji zadužuje knjigu. Ona vrši registraciju da je navedena knjiga zadužena kod navedenog korisnika. U slučaju da je evidencijski broj knjige ili članski broj korisnika neispravan, ili ukoliko je knjiga već zadužena, metoda baca izuzetak tipa "`logic_error`" uz prateće tekstove "Knjiga nije nadjena", "Korisnik nije nadjen", odnosno "Knjiga vec zaduzena" (prvo se testira knjiga pa korisnik, tako da ukoliko nisu ispravni ni evidencijski broj knjige niti članski broj korisnika, prijavljuje se prvi tekst). Metoda "`RazduziKnjigu`" prima kao parametar evidencijski broj knjige. Ona registrira da knjiga više nije zadužena. U slučaju da je evidencijski broj neispravan, ili ukoliko knjiga uopće nije bila zadužena, metoda baca izuzetak tipa "`logic_error`" uz prateće tekstove "Knjiga nije nadjena" odnosno "Knjiga nije zaduzena". Konačno, metoda "`PrikaziZaduzenja`" prima kao parametar članski broj korisnika, a vrši ispis podataka o svim knjigama koje je zadužio navedeni korisnik, na isti način kao u metodi "`IzlistajKnjige`" (bez prikazivanja ko je zadužio knjigu). U slučaju da korisnik nije zadužio niti jednu knjigu, metoda ispisuje tekst "Nema zaduzenja za tog korisnika!", dok u slučaju da je članski broj neispravan, metoda baca izuzetak tipa "`logic_error`" uz prateći tekst "Korisnik nije nadjen".

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

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

3. Jedan od nedostataka jezika opće namjene kao sto je C++ u odnosu na specijalističke matematski orjentirane jezike je nepostojanje podrške za egzaktni rad sa racionalnim brojevima (razlomcima) kakav postoji npr. u programskim jezicima Wolfram (na kojem je zasnovan programski paket Wolfram Mathematica) ili Maple. Srećom, koncept klasa u jeziku C++ omogućava korisniku da sam definira nove tipove podataka koji zadovoljavaju njegove potrebe. Tako je moguće definirati novi tip podataka (klasu) koji omogućava tačan rad sa razlomcima (umjesto njihovog približnog predstavljanja preko decimalnih brojeva). Nažalost, takav tip podataka nije lako napraviti da radi posve dobro. Najveći problem je što tačan rezultat čak i vrlo prostih aritmetičkih operacija sa razlomcima čiji brojnici i nazivnici uopće nisu osobito veliki može zahtijevati izuzetno velike brojeve za reprezentaciju brojnika i nazivnika. Na primjer, već u računu

$$\frac{3728}{14611}+\frac{9823}{23717}+\frac{7465}{19434}=\frac{7094377705241}{6734446276758}$$

niti brojnik niti nazivnik rezultata na većini računara ne mogu da stane u tip "`int`" iako je sam rezultat sasvim "normalne" veličine (približno oko 1.05345). Istina je da se ovaj rezultat može

aproksimirati razlomkom 133380/126613 sa greškom manjom od 10^{-12}, ili recimo razlomkom 397630891/377457190 sa greškom manjom od 10^{-18} (što je izvanredna tačnost), činjenica je da ovo ipak *nisu tačni rezultati*, pa se gubi cijela poenta. Naravno, situacija se popravlja ukoliko se odlučimo da brojnik i nazivnik čuvamo u nekom širem cjelobrojnom tipu, poput "`long long int`", ali i to ima svoje granice: brojnik i nazivnik rezultata

$$\frac{3728}{14611}+\frac{9823}{23717}+\frac{7465}{19434}+\frac{8606}{14243}+\frac{4931}{22067}=\frac{3981669383975848685651}{2116638357164443168998}$$

na većini računara ne mogu stati ni u tip "`long long int`" (ovaj rezultat je približno 1.88113). I dalje je činjenica da se ovaj rezultat može veoma dobro aproksimirati jednostavnijim razlomcima (recimo razlomkom 552123131/293506287 sa greškom manjom od 10^{-17}), ali ponovo to je bijeg od osnovnog cilja koji želimo postići – da imamo tačne rezultate a ne aproksimacije.

Opisani problem rješava se na razne načine. Jedino pravo rješenje je da se brojnik i nazivnik uopće ne čuvaju u cjelobrojnim varijablama, nego da se individualne cifre brojnika i nazivnika čuvaju recimo u stringu ili vektoru. Mada se time rješava opisani problem, takva izvedba je vrlo komplicirana, jer se sve računske operacije moraju izvoditi "pješke" (kao pri ručnom izvođenju aritmetičkih operacija), cifru po cifru. Nažalost, internet je prepun loših realizacija ovakve klase, koje koriste algoritam noja – gurni glavu u pijesak (tj. ignoriraj problem, u nadi da će i problem ignorirati tebe). Kod takvih realizacija, ukoliko brojnik i nazivnik rezultata ne mogu da stanu u predviđeni tip za njihovo smještanje, rezultat je jednostavno *pogrešan*, pri čemu korisnik nema nikakvu indikaciju da je rezultat pogrešan. Recimo, kod velikog broja loših interpretacija koje koriste tip "`int`", račun

$$\frac{396679}{437786}+\frac{147419}{131439}=\frac{116677065415}{57542154054}$$

kod kojeg brojnik i nazivnik očito nisu reprezentabilni u tipu "`int`", daje na većini računarskih arhitektura pogrešan rezultat 712948423/1707579206 (ovaj rezultat je posljedica toga kako se na većini računarskih arhitektura tretira prekoračenje). Ovaj rezultat je zaista vrlo pogrešan, jer mu je vrijednost približno 0.41752, dok tačan rezultat približno iznosi 2.02768. Nažalost, korisnik takvih klasa (kojih je, da ponovimo, internet prepun), nema nikakvu indikaciju da je rezultat posve neupotrebljiv.

U ovom zadatku Vi ćete uraditi nešto bolje. Razvićete klasu "`Razlomak`" kod koje će se brojnik i nazivnik čuvati u atributima tipa "`long long int`" (dakle, i dalje će biti mogući problemi sa reprezentacijom rezultata), ali će sve aritmetičke operacije bacati izuzetak ukoliko rezultat nije egzaktno reprezentabilan, pa će, ako ništa drugo, korisnik klase barem znati da za taj račun sa tom klasom ne može dobiti tačan rezultat (kasnije će biti opisano kako ovo tačno postići). Mogu se naći i bolje implementacije, koje u slučaju da rezultat nije egzaktno reprezentabilan vraćaju najpribližniji reprezetnabilni razlomak (tj. takav da je greška što je god moguće manja), uz mogućnost da korisnik sazna da li je dobio egzaktan rezultat ili ne. Mada bi bilo veoma lijepo napraviti i takvu klasu, za njenu implementaciju potrebno je poznavati neke tehnike iz teorije brojeva (verižne razlomke, itd.) koje ne spadaju u predznanja studenata koji će raditi ovaj zadatak.

Klasa "`Razlomak`" treba da sadrži dva privatna atributa koji čuvaju vrijednost brojnika i nazivnika razlomka. Konstruktor klase ima dva parametra, koji respektivno predstavljaju vrijednost brojnika i nazivnika razlomka. Ovaj konstruktor treba automatski da izvrsi skraćivanje eventualnih zajedničkih faktora u brojniku i nazivniku, tako da ce deklaracija poput

```
Razlomak r(10, 15);
```

dovesti do toga da ce brojnik razlomka "`r`" biti 2, a nazivnik 3. Također, konstruktor treba da obezbijedi da u slučaju da je razlomak negativan, negativni znak bude zapamćen u brojniku, a ne u nazivniku. Drugim riječima, nakon deklaracije

```
Razlomak r(2, -3);
```

vrijednost brojnika treba da bude –2, a nazivnika 3. Naravno, nakon deklaracije poput

```
Razlomak r(-2, -3);
```

vrijednosti brojnika i nazivnika treba da budu 2 i 3, jer se negativni znak krati. U slučaju da se za vrijednost nazivnika zada nula, konstruktor treba da baci izuzetak tipa "`logic_error`" uz prateći tekst "Nekorektan razlomak". Kraćenje izvršite dijeljenjem brojnika i nazivnika sa njihovim najvećim zajedničkim djeliocem (NZD). Za računanje NZD postoji mnogo loših i dobrih metoda, a jedan od najboljih (jednostavan i ujedno vrlo efikasan) je Euklidov algoritam, koji se zasniva na rekurzivnoj formuli

$$\text{NZD}(p,q) = \begin{cases} p, \text{za } q = 0 \\ \text{NZD}(q, \text{mod}(p,q)), \text{za } q \neq 0 \end{cases}$$

Ovdje "mod" označava ostatak pri dijeljenju. Ova formula se trivijalno može prevesti u rekurzivnu funkciju za računanje NZD, a podjednako je jednostavno izvesti stvari i bez rekurzije (u petlji se par (p, q) zamjenjuje sa parom $(q, \text{mod}(p,q))$ dok q ne postane 0, p je tada traženi NZD). U svakom slučaju, pošto će računanje NZD trebati u ovom zadatku na više mjesta, njegovo računanje povjerite privatnoj statičkoj funkciji.

Konstruktor treba također da ima podrazumijevane parametre koji omogućavaju da se razlomak zada samo jednim parametrom ili sasvim bez parametara. Ukoliko se zada sa samo jednim parametrom, taj parametar predstavlja brojnik, a nazivnik se tada postavlja na jedinicu. Treba omogućiti da se putem ovog konstruktora vrši i automatska pretvorba cijelih brojeva u objekte tipa "`Razlomak`". Konačno, ukoliko se parametri skroz izostave, kreira se nul-razlomak (brojnik 0, nazivnik 1). U svim slučajevima, mora biti podržano da se umjesto okruglih mogu koristiti i vitičaste zagrade, tj. da se može pisati "`Razlomak r{2, 3}`" umjesto "`Razlomak r(2, 3)`" odnosno "`Razlomak r{2}`" umjesto "`Razlomak r(2)`". Predvidite i trivijalne pristupne metode "`DajBrojnik`" i "`DajNazivnik`" koje vraćaju vrijednost brojnika odnosno nazivnika respektivno.

Najvažniji dio klase su operatorske funkcije za podršku aritmetičkih operacija. Na prvom mjestu, treba podržati binarne operatore "`+`", "`-`", "`*`" i "`/`" za četiri osnovne aritmetičke operacije. Pri tome, nemojte za tu svrhu koristiti dobro poznate formule

$$\frac{p_1}{q_1} \pm \frac{p_2}{q_2} = \frac{p_1 \cdot q_2 \pm p_2 \cdot q_1}{q_1 \cdot q_2}, \qquad \frac{p_1}{q_1} \cdot \frac{p_2}{q_2} = \frac{p_1 \cdot p_2}{q_1 \cdot q_2}, \qquad \frac{p_1}{q_1} \Big/ \frac{p_2}{q_2} = \frac{p_1 \cdot q_2}{q_1 \cdot p_1}$$

Naime, mada u ove formule neosporno tačne, postoji velika mogućnost da će ove formule dovesti do prekoračenja, čak i u slučaju kada je rezultat savršeno reprezentabilan nakon što se brojnik i nazivnik skrate. Mnogo manja vjerovatnoća prekoračenja je ako se koriste formule

$$\frac{p_1}{q_1} \pm \frac{p_2}{q_2} = \frac{p_1 \cdot (q_2/r) \pm p_2 \cdot (q_1/r)}{q_1 \cdot (q_2/r)}, \qquad \frac{p_1}{q_1} \cdot \frac{p_2}{q_2} = \frac{(p_1/s) \cdot (p_2/t)}{(q_1/t) \cdot (q_2/s)}, \qquad \frac{p_1}{q_1} \Big/ \frac{p_2}{q_2} = \frac{(p_1/u) \cdot (q_2/r)}{(q_1/r) \cdot (p_2/u)}$$

gdje je $r = \text{NZD}(q_1, q_2)$, $s = \text{NZD}(p_1, q_2)$, $t = \text{NZD}(p_2, q_1)$ i $u = \text{NZD}(p_1, p_2)$. Pri tome, svi podizrazi poput q_2/r itd. su *cijeli brojevi* (jer je q_2 djeljiv sa r). Na primjer, po klasičnoj formuli imamo

$$\frac{2337}{3740} + \frac{4014}{5225} = \frac{2337 \cdot 5225 + 4014 \cdot 3740}{3740 \cdot 5225} = \frac{27223185}{19541500} = \frac{44997}{32300} \quad \text{(nakon kraćenja)}$$

dok po modificiranoj formuli imamo:

$$\frac{2337}{3740} + \frac{4014}{5225} = \frac{2337 \cdot (5225/55) + 4014 \cdot (3740/55)}{3740 \cdot (5225/55)} = \frac{494967}{355300} = \frac{44997}{32300} \quad \text{(nakon kraćenja)}$$

jer je $\text{NZD}(3740, 5225) = 55$. Vidimo da se modificirane formule daju manje međurezultate, što smanjuje opasnost od prekoračenja. Ipak, ni ove modificirane formule nemaju garanciju da neće doći do prekoračenja čak i u slučajevima kada bi rezultat bio savršeno reprezentabilan (nakon kraćenja), ali ovo je maksimum koji se može postići bez korištenja komplikovanije matematike. Dakle, ovo su formule koje ćete koristiti za realizaciju aritmetičkih operacija. U svakom slučaju, rezultati svake od operacija uvijek treba da budu do kraja skraćeni (što ćete najlakše postići tako što ćete brojnik i nazivnik željenog rezultata prosto proslijediti konstruktoru, koji će obaviti neophodno skraćivanje). Ukoliko bilo koje od računanja dovede do prekoračenja u bilo kojem

međurezultatu, treba baciti izuzetak tipa "`overflow_error`" uz prateći tekst "Nemoguce dobiti tacan rezultat". Ostaje pitanje kako detektirati prekoračenje. Nažalost, prekoračenje je nemoguće detektirati nakon što se već desi, moguće je jedino unaprijed utvrditi da će doći do prekoračenja. Sada ćemo objasniti i kako. Neka su m i M najmanja odnosno najveća vrijednost koja može stati u neki tip podataka ("`long long int`" u našem slučaju). Ako je y pozitivan, računanje $x + y$ će izazvati prekoračenje ako i samo ako je $x > M - y$, a ako je y negativan, do prekoračenja će doći ako i samo ako je $x < m - y$. Oduzimanje se svodi na slično razmatranje, samo uz suprotan tretman znaka y (jer je $x - y = x + (-y)$). Što se tiče množenja, ako je y pozitivan, do prekoračenja će doći ako i samo ako je $x < m / y$ ili $x > M / y$, a ako je y negativan, do prekoračenja će doći ako i samo ako je $x = m$ ili $-x > M / (-y)$ ili $-x < m / (-y)$. Ovdje je pretpostavljeno da je $m = -M - 1$, što je ispunjeno na gotovo svim "normalnim" raćunarskim arhitekturama. Napomenimo da prividno ekvivalentni izrazi poput $x < \mathrm{M} / y$ i $x > m / y$ mogu napraviti problem, recimo za $\mathrm{y} = -1$ izraz m / y dovodi do prekoračenja (jer je $-m > M$).

Pored ova četiri binarna operatora, treba podržati i unarne operatore "`+`" i "`-`". Unarni operator "`-`" vraća razlomak na koji je primijenjen sa izvrnutim znakom (uz bacanje analognog izuzetka kao u slučaju binarnih operatora ukoliko rezultat negacije nije reprezentabilan), dok unarni operator "`+`" vraća svoj operand neizmijenjen. U skladu sa programerskim bontonom, potrebno je podržati i prekopljene operatore "`+=`", "`-=`", "`*=`" i "`/=`", pri čemu izrazi "`r1 += r2`", "`r1 -= r2`", "`r1 *= r2`" i "`r1 /= r2`" treba da budu semantički ekvivalentni izrazima poput "`r1 = r1 + r2`", "`r1 = r1 + r2`", "`r1 = r1 * r2`" i "`r1 = r1 / r2`". Također, treba podržati i unarne operatore "`++`" i "`--`", pri čemu izrazi "`r++`" i "`r--`" efektivno djeluju kao izrazi "`r += 1`" i "`r -= 1`". Pri tome treba podržati kako prefiksnu, tako i postfiksnu verziju ovih operatora (postfiksne verzije vraćaju kao rezultat vrijednost razlomka prije obavljene izmjene).

Za poređenje objekata tipa "`Razlomak`" treba podržati relacione operatore "`<`", "`>`", "`<=`", "`>=`","`==`" i "`!=`". Tu također treba biti oprezan kako to izvesti. Iz matemtike je poznato da ukoliko su p_1/q_1 i p_2/q_2 dva razlomka (pri čemu je $q_1 > 0$ i $q_2 > 0$), uvjet poput $p_1/q_1 \geq p_2/q_2$ sa matematičkog aspekta ekvivalentan je uvjetu $p_1 \cdot q_2 \geq p_2 \cdot q_1$, itd. Međutim, veliki je problem što pri računanju proizvoda $p_1 \cdot q_2$ i $p_2 \cdot q_1$ može doći do prekoračenja. Mada postoje i ovdje alternativne formule (recimo, može se koristiti ekvivalentni uvjet $(p_1/r) \cdot (q_2/s) \geq (p_2/r) \cdot (q_1/s)$ uz $r = \mathrm{NZD}(p_1, p_2)$ i $s = \mathrm{NZD}(q_1, q_2)$ kod kojeg je vjerovatnoća prekoračenja mnogo manja), za potrebe ovog zadatka sasvim dobro rješenje je izračunati *aproksimativne decimalne vrijednosti* razlomaka p_1/q_1 i p_2/q_2 (dijeljenjem brojnika sa nazivnikom) i uporediti dobijene decimalne vrijednosti. Pri tome, za dijeljenje treba koristiti tip "**`long double`**", jer samo on garantira dovoljnu preciznost da se ovo pouzdano obavi. Ipak, ovo rješenje nije univerzalno, jer na računarskim arhitekturama na kojima tip "`long long int`" ima više od 64 bita (tako da je moguće zapamtiti i brojeve sa više od 20 cifara), moguće je imati dva različita razlomka, ali čije su aproksimativne decimalne vrijednosti toliko bliske da se čak ni uz pomoć tipa "**`long double`**" ne može detektirati razlika.

Preklopljeni operator "`<<`"treba da podrži ispis objekata tipa "`Razlomak`" na ekran. Ukoliko je p brojnik a q nazivnik razlomka koji se ispisuje, ispis treba da izgleda kao p/q, osim ukoliko je nazivnik jednak jedinici, kada se ispisuje samo vrijednost brojnika. Slično, preklopljeni operator "`>>`" treba podrati unos objekata tipa "`Razlomak`" sa tastature. Potrebno je omogućiti da se ovi objekti unose u formi p/q, gdje su p i q cijeli brojevi, ali treba omogućiti da se unese i obični cijeli broj (u tom slučaju se podrazumijeva da je nazivnik jedinica). U slučaju neispravnog unosa, ulazni tok treba da dospije u neispravno stanje.

Konačno, treba podržati i automatsku konverziju objekata tipa "`Razlomak`" u tip "**`long double`**", koji omogućava upotrebu promjenljivih tipa "`Razlomak`" u kontekstima gdje se očekuju realni brojevi, tj. promjenljive tipa "**`long double`**" (treba znati da će ovim biti automatski podržana i konverzija u tip "**`double`**", pri čemu će ako se zahtijeva konverzija u tip "**`double`**", dodatne decimale koje nudi tip "**`long double`**" biti prosto odbačene). Rezultat konverzije je približna decimalna vrijednost koja se dobija dijeljenjem brojnika sa nazivnikom.

Sve metode obavezno implementirajte izvan deklaracije klase, osim trivijalnih metoda koje trebate implementirati direktno unutar deklaracije klase. Sve metode koje su inspektori, obavezno deklarirajte kao takve. Također, obavezno napišite i mali testni program u kojem čete testirati sve funkcionalnosti napisane klase.

4. U računarskoj grafici veoma se često javlja potreba za radom sa matricama (pomoću njih se vrlo efektno modeliraju razne geometrijske transformacije u ravni i prostoru), pri čemu te matrice nikada nemaju format veći od 4×4, pri čemu su mogući su i manji formati (stoga veliki broj današnjih procesora direktno hardverskim putem podržava manipulacije sa matricama formata do 4×4, upravo radi podrške računarskoj grafici). Radi lakšeg rada sa ovakvim matricama, praktično je razviti posebnu klasu, koju možemo nazvati recimo "`GMatrica`". Kako se u raznim primjenama tip elemenata takvih matrica može razlikovati (nekada su dovoljni cijeli brojevi, nekad su potrebi realni, a nekada čak i kompleksni brojevi), ovu klasu bi zgodno bilo realizirati kao generičku, tako da se tip elemenata može specificirati. Vaš zadatak je da razvijete upravo takvu klasu, u skladu sa opisom koji slijedi. S obzirom da su maksimalne dimenzije takvih matrica ograničene na 4×4, nema potrebe za korištenjem dinamičke alokacije memorije, niti fleksibilnih tipova podataka kao što su vektori (odnosno vektori vektôrā). Stoga elemente matrice čuvajte u atributu koji je obični dvodimenzionalni niz neodređenog tipa elemenata (koji se naknadno zadaje) formata 4×4. Međutim, kako format matrice može biti i manji od 4×4, predvidite i dva atributa koji čuvaju stvarni broj redova i kolona matrice (naravno, u slučaju kada je format manji od 4×4, neki elementi dvodimenzionalnog niza pohranjenog unutar odgovarajućeg atributa biće neiskorišteni).

   Generička klasa "`GMatrica`" treba da sadrži nekoliko konstruktora. Na prvom mjestu, tu je konstruktor bez parametara, koji kreira praznu "matricu" formata 0×0 (u suštini, jedini razlog za postojanje ovog konstruktora je da dozvoli mogućnost kreiranja nizova čiji su elementi tipa "`GMatrica`"). Zatim, tu je konstruktor sa tri parametra, koji redom predstavljaju broj redova, broj kolona i vrijednost na koju se inicijaliziraju svi elementi matrice (treći parametar treba da bude u skladu sa tipom elemenata matrice). Posljednji parametar ovog konstruktora treba da ima podrazumijevanu vrijednost koja je podrazumijevana vrijednost tipa elemenata matrice (npr. 0 ako su elementi brojčane prirode), i ona se koristi ukoliko se parametar izostavi. U slučaju da broj redova ili broj kolona nisu legalni (u opsegu od 0 do 4, u skladu sa postavkom zadatka), treba baciti izuzetak tipa "`logic_error`" uz prateći tekst "Ilegalan format matrice". Dalje, treba podržati generički konstruktor koji prima parametar također tipa "`GMatrica`", ali sa drugačijim tipom elemenata. Inače, konstruktori i funkcije članice također mogu biti generički (to se koristi recimo ukoliko je tip parametara djelimično ili potpuno nepoznat), mada to nije eksplicitno naglašeno na predavanju. U tom slučaju, "**`template`**" deklaraciju treba staviti ispred deklaracije ili definicije konstruktora odnosno funkcije članice, kao i u slučaju običnih generičkih funkcija. Ovaj konstruktor bi trebao da omogući recimo da se matrica (tj. objekat tipa "`GMatrica`") čiji su elementi tipa "**`double`**" može inicijalizirati matricom čiji su elementi recimo tipa "**`int`**" (naravno, tako nešto može raditi samo ukoliko je tip elemenata izvorne matrice konvertibilan u tip elemenata odredišne matrice). Ovaj konstruktor će indirektno (putem automatske prevrobe tipa) omogućiti i međusobno dodjeljivanje matrica različitih tipova elemenata. Sljedeći je generički konstruktor sa jednim parametrom koji je klasični (C-ovski) dvodimenzionalni niz formata 4×4 čiji tip elemenata odgovara tipu elemenata odredišne matrice, ili je konvertibilan u tip elemenata odredišne matrice. Ovaj konstruktor prosto kreira objekat tipa "`GMatrica`" koji modelira matricu formata 4×4 i inicijalizira je odgovarajućim vrijednostima iz dvodimenzionalnog niza koji je zadan kao parametar. Glavna uloga ovog konstruktora je da omogući automatsku konverziju dvodimenzionalnih nizova formata 4×4 u objekte tipa "`GMatrica`". Zatim, treba imati i konstruktor koji prima jedan parametar tipa vektora vektôrā, čiji su elementi istog tipa kao što je pretpostavljeni tip elemenata odredišne matrice. Ovaj konstruktor kreira matricu na osnovu elemenata vektora vektôrā koji mu je proslijeđen kao parametar. Njegova osnovna namjena je ponovo da podrži automatsku pretvorbu vektora vektôrā u odgovarajući tip "`GMatrica`". Uvjet da ova inicijalizacija bude ispravna je da broj redova vektora vektôrā bude od 0 do 4, te da svi redovi imaju isti broj elemenata, koji nije veći od 4. U suprotnom treba baciti izuzetak tipa "`logic_error`" uz prateći tekst "Ilegalan format matrice". Primijetimo da će zbog postojanja automatske konverzije inicijalizacionih listi u vektore, ovaj konstruktor automatski podržati i konstrukcije poput "`GMatrica<int> m({{1, 2}, {3, 4}})`". Konačno, posljednji konstruktor je sekvencijski konstruktor koji će podržati kreiranje objekata tipa "`GMatrica`" direktno iz inicijalizacijskih listi, tj. koji će omogućiti konstrukcije poput "`GMatrica<double> m{{1, 2}, {3, 4}}`" (primijetite da u ovom slučaju, za razliku od prethodnog konstruktora, ovdje nema dodatnih okruglih zagrada). Slično prethodnom konstruktoru, i ovaj konstruktor baca izuzetak ukoliko nije zadana ispravna forma matrice. Sve konstruktore *obavezno treba implementirati izvan deklaracije klase*!

Većina funkcionalnosti klase "GMatrica" će se postizati putem operatora, tako da od metoda ova klasa treba podržati samo pristupne metode "DajBrojRedova" i "DajBrojKolona" koje daju broj redova odnosno broj kolona matrice. Od operatora, tu su na prvom mjestu binarni operatori "+", "-" i "*" koji redom nalaze zbir, razliku i proizvod matrica, pod uvjetom da su matrice međusobno saglasne za izvođenje tih operacija. U suprotnom, treba baciti izuzetak tipa "domain_error" uz prateći tekst "Nedozvoljena operacija". Matrice ne moraju biti istog tipa elemenata (recimo, može se sabrati objekat tipa "GMatrica" čiji su elementi tipa "int" sa sličnim objektom čiji su elementi tipa "**double**"), pri čemu rezultirajuća matrica treba imati onakav tip elemenata kakav se dobije primjenom odgovarajuće operacije nad elementima jedne i druge matrice (ovdje će Vam trebati "**decltype**" operator). Operator "*" također treba da podržava množenje broja sa matricom odnosno matrice sa brojem. Tip broja ne mora biti isti kao tip elemenata matrice. Odgovarajuće operatorske funkcije *ne treba da budu funkcije članice* i *obavezno ih treba implementirati izvan deklaracije klase* (upadnete li u probleme, a najvjerovatnihe hoćete, informacije date pred kraj Predavanja 11_b mogu Vam biti od koristi). Dalje, treba podržati prekopljene binarne operatore "+=","-=" i "*=" takve da izrazi poput "X += Y", "X -= Y" i "X *= Y" uvijek imaju isti efekat kao i izrazi "X = X + Y", "X = X - Y" i "X = X * Y" kad god to ima smisla. Pri tome, ove operatore *obavezno izvedite putem funkcija članica*, koje trebaju biti implementirane *izvan deklaracije klase*, pri čemu se njihova realizacija ne smije trivijalno svoditi na operatore "+", "-" i "*" (tj. nije dozvoljena izvedba operatora "+=" koja se prosto sastoji od naredbe poput "**return** ***this** = ***this** + y"). Binarni operatori "==" i "!=" koji daju rezultat "**true**" ukoliko su operandi jednake odnosno različite matrice, odnosno "**false**" u suprotnom. Dalje, treba podržati operator "()" koji omogućava pristup elementima matrice zadavanjem reda i kolone putem parametara u zagradi, npr. "m(2, 3)" je element u drugom redu i trećoj koloni. Indeksi se gledaju "matematički", dakle od jedinice a ne od nule. Ukoliko indeksi nisu u dozvoljenom opsegu (u skladu sa trenutnim brojem redova i kolona), treba baciti izuzetak tipa "range_error" uz prateći tekst "Nedozvoljen indeks". Rezultat treba biti referenca na element, tako da se može koristiti i sa lijeve strane operatora dodjele, osim u slučaju kada se ovaj operator primijeni na konstantnu matricu (tada umjesto reference treba vratiti kopiju elementa). Alternativno, pristup mora biti omogućen i putem operatora "[]", sintaksom u C/C++ stilu poput "m[1][1]", pri čemu u tom slučaju indeksacija ide *od nule* i *bez provjere ispravnosti indeksa*. I u ovom slučaju potrebno je imati konzistentan tretman konstantnih matrica. Na kraju, treba podržati i operatore za rad sa tokovima. Preklopljeni operator "<<"omogućava ispis matrice na izlazni tok. Elementi matrice se ispisuju red po red, pri čemu se za ispis svakog elementa onoliki prostor koliki je postavljen pomoću poziva funkcije "cout.width" ili pomoću manipulatora "setw". Na primjer, ukoliko je "a" objekat tipa "GMatrica", prilikom ispisa "cout << setw(10) << a" za ispis elemenata matrice treba rezervirati prostor od 10 mjesta. Ukoliko je zadata širina manja od 6 mjesta, treba koristiti podrazumijevanu minimalnu širinu od 6 mjesta. Konačno, preklopljeni operator ">>" omogućava unos matrice sa ulaznog toka. Elementi matrice se nalaze unutar uglastih zagrada a međusobno se razdvajaju zarezima, dok tačka-zarez označava prelazak u sljedeću kolonu. Na primjer, matrica formata 2×3 čiji su elementi u prvom redu 2, 5 i 4, a elementi u drugom redu 3, 0 i -2 unosi se sa tastature kao "[2,5,4;3,0,-2]". U slučaju bilo kakvih grešaka prilikom unosa (što uključuje i neusklađen broj unijetih elemenata sa dimenzijama matrice), ulazni tok treba da se postavi u neispravno stanje.

Destruktor, kopirajući i pomjerajući konstruktori te preklopljeni operatori dodjele nisu potrebni za ispravno funkcioniranje klase, s obzirom da se svi elementi matrice čuvaju u običnom nizu, tj. svi elementi su kompletno sadržani u atributima klase. Obavezno napišite i testni program u kojem ćete demonstrirati sve elemente napisane generičke klase.

5. Meteorološka stanica čuva evidenciju o minimalnoj i maksimalnoj dnevnoj temperaturi u toku nekog perioda posmatranja u klasi nazvanoj "Temperature". Vaš zadatak je da implementirate tu klasu. Minimalne i maksimalne temperature se čuvaju respektivno u dva privatna atributa koji su liste cijelih brojeva, realizirane pomoću bibliotečke generičke klase "list"). Konstruktor klase "Temperature" zahtijeva dva parametra koji redom predstavljaju minimalnu i maksimalnu temperaturu (po iznosu) koja se može registrirati, pri čemu se baca izuzetak tipa "range_error" uz prateći tekst "Nekorektne temperature" ukoliko je minimalna dozvoljena temperatura veća od maksimalne. Metoda "RegistrirajTemperature" prima kao parametar uređeni par (tipa "pair") cijelih brojeva, koji respektivno predstavljaju minimalnu i maksimalnu dnevnu temperaturu, i upisuje ih u odgovarajuće liste. Pri tome se baca izuzetak tipa "range_error" uz prateći tekst "Nekorektne temperature" ukoliko je bilo koja od temperatura manja od minimalno dozvoljene ili veća od maksimalno dozvoljene, ili ukoliko je minimalna dnevna temperatura veća od maksimalne.

Metoda "BrisiSve" briše sve unesene temperature, dok metoda "BrisiNegativneTemperature" briše samo registrirane parove temperatura kod kojih su obje temperature negativne (ukoliko takvih parova nema, ne dešava se ništa). Metoda "DajBrojRegistriranihTemperatura" daje broj registriranih parova temperatura. Treba podržati i metode "DajMinimalnuTemperaturu", "DajMaksimalnuTeperaturu", "DajBrojTeperaturaVecihOd" te "DajBrojTeperaturaManjihOd". Prve dvije metode vraćaju kao rezultat minimalnu i maksimalnu temperturu registriranu u toku perioda posmatranja, dok treća i četvrta vraćaju broj dana u kojima je registrirana temperatura veća odnosno manja od zadane. Sve četiri metode bacaju izuzetak tipa "logic_error" uz prateći tekst "Nema registriranih temperatura" u slučaju da nema registriranih temperatura. Za realizaciju svih ovih metoda nije dozvoljeno koristiti petlje, niti lambda funkcije ili pomoćne imenovane funkcije kriterija, nego isključivo samo odgovarajuće funkcije i/ili funktore iz biblioteka "algorithm" i "functional". S obzirom da ćete za tu svrhu morati koristiti veznike, a na njihovu upotrebu treba se navići, prvo probajte riješiti problem uz pomoć lambda funkcija. Kada Vam funkcija proradi, probajte istu funkcionalnost postići pomoću jednostavnijih ali zastarjelih veznika "bind1st" ili "bind2st". Kada Vam i to proradi, zamijenite zastarjele veznike sa boljim i univerzalnijim veznikom "bind" (zastarjele verzije veznika nemojte ostavljati u završnoj verziji).

U klasi "Temperature" potrebno je podržati i neke operatore, pri čemu ni u jednom slučaju nije dozvoljeno koristiti petlje, niti pomoćne imenovane ili lambda funkcije, nego isključivo funkcije i/ili funktore iz biblioteka "algorithm" i "functional" (vrijede iste napomene kao i gore). Unarni operator "!" primijenjen na objekat daje logičku vrijednost "tačno" ukoliko nema niti jedne registrirane temperature, u suprotnom daje logičku vrijednost "netačno". Unarni operatori "++" i "--"povećavaju sve minimalne odnosno smanjuju sve maksimalne dnevne temperature za jedinicu. Ukoliko pri tome neka od minimalnih dnevnih temperatura premaši odgovarajuću maksimalnu, treba baciti izuzetak tipa "logic_error" uz prateći tekst "Ilegalna operacija". Potrebno je podržati kako prefiksnu, tako i postfiksnu verziju ovih operatora. Unarni operator "*" primijenjen na objekat tipa "Temperature" daje kao rezultat vektor čiji su elementi razlike između maksimalnih i minimalnih registriranih dnevnih temperatura, dok unarni operatori "-" i "+" daju kao rezultat vektore koji respektivno daju koliko su minimalne dnevne temperature bile veće od minimalno dozvoljene vrijednosti, odnosno koliko su maksimalne dnevne temperature bile manje od maksimalno dozvoljene vrijednosti. Preklopljeni operator "[]" omogućava da se direktno pročita *i*-ti registrirani par temperatura (numeracija ide od jedinice, a vraća se objekat tipa "pair"). Ukoliko je indeks izvan dozvoljenog opsega, treba baciti izuzetak tipa "range_error" uz prateći tekst "Neispravan indeks". Pri tome se ovaj operator *ne može koristiti* za izmjenu pohranjenih podataka, odnosno ne može se koristiti sa lijeve strane znaka jednakosti. Preklopljeni binarni operatori "+" i "-" djeluju na sljedeći način. Ukoliko je "X" objekat tipa "Temperature", a "Y" cijeli broj, tada je "X + Y" je novi objekat tipa "Temperature" u kojem su sve registrirane temperature (i minimalne i maksimalne) povećane za iznos "Y". Slično, "X - Y" je novi objekat tipa "Temperature" u kojem su sve registrirane temperature umanjene za iznos "Y". Pri istim tipovima operanada "X" i "Y", izraz "Y + X" treba da ima isto značenje kao i izraz "X + Y", dok izraz "Y - X" predstavlja novi objekat u kojem su sve registrirane temperature oduzete od vrijednosti "Y", uz zamjenu uloge minimalnih i maksimalnih dnevnih temperatura. Ukoliko se kao rezultat bilo koje od ovih operacija dogodi da neka od temperatura izađe izvan dozvoljenog opsega, treba baciti izuzetak tipa "logic_error" uz prateći tekst "Prekoracen dozvoljeni opseg temperatura". U svim ostalim situacijama, izrazi "X + Y" odnosno "X - Y" ne trebaju biti sintaksno ispravni. Dalje, potrebno je podržati preklopljene operatore "+=" i "-=" čiji je cilj da značenje izraza oblika "X += Y" odnosno "X -= Y" bude identično značenju izraza "X = X + Y" i "X = X - Y" kad god oni imaju smisla. Za poređenje objekata tipa "Temperature" treba podržati relacione operatore "==" i "!=" koji ispituju da li su dva objekta tipa "Temperature" identična ili nisu, pri čemu se smatra da su dva objekta ovog tipa jednaka ukoliko se poklapaju u svim aspektima (jednak broj registriranih temperatura, jednake registrirane temperature, i jednaki dozvoljeni opsezi temperatura). Konačno, potrebno je podržati prekopljeni operator "<<" koji ispisuje sve registrirane temperature, minimalne u jednom redu, a zatim maksimalne u drugom redu, pri čemu su temperature unutar jednog reda međusobno razdvojene razmakom (i ovo treba izvesti bez petlji i pomoćnih funkcija).

Sve metode implementirajte izvan klase, osim metoda čija je implementacija dovoljno kratka, u smislu da zahtijeva recimo jednu ili dvije naredbe. Metode koje su po prirodi inspektori obavezno treba deklarirati kao takve. Obavezno napišite i mali testni program u kojem će se testirati sve elemente napisane klase.