Zadaća 5.

Ova zadaća nosi ukupno 4 poena, pri čemu prvi i trugi zadatak nose po 1 poen, treći zadatak nosi 0,7 poena a četvrti zadatak 1,3 poena. Sva četiri zadatka se mogu uraditi na osnovu gradiva sa prvih 12 predavanja i pretpostavljenog predznanja iz predmeta "Osnove računarstva". Rok za predaju ove zadaće je nedjelja, 8. VI 2014. (do kraja dana) i ne može se produžiti. Zadaće se predaju putem Zamgera.

1. 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 četiri parametara koji inicijalizira sve atribute na vrijednosti zadane parametrima (redoslijed parametara je isti kao i redoslijed gore navedenih atributa), zatim odgovarajuće trivijalne pristupne metode "`DajClanskiBroj`", "`DajImeIPrezime`", "`DajAdresu`" i "`DajTelefon`" koje prosto vraćaju vrijednosti odgovarajućih atributa, te metodu "`Ispisi`" koja ispisuje podatke o korisniku na ekran. Ispis treba da izgleda ovako:

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

   Klasa "`Film`" sadrži privatne atribute koji čuvaju informacije o evidencijskom broju video trake ili CD-a na kojem je film, zatim da li je film na video traci ili DVD-u, nazivu filma, žanru i godini produkcije, kao i informaciju o eventualnom zaduženju filma. Evidencijski broj i godina izdavanja su cijeli brojevi, informacija da li je film na video traci ili DVD-u je logičkog tipa, informacija o zaduženju čuva se kao pokazivač (obični) na korisnika koji je zadužio film odnosno nul-pokazivač ukoliko film nije zadužen, dok su ostali atributi stringovnog tipa. Interfejs klase sadrži konstruktor 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 (svi izuzeci koji se spominju u ovom zadatku trebaju biti tipa "`logic_error`" uz prikladan prateći tekst koji možete sami odabrati). 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 tačno 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.

2. Definirajte i implementirajte klasu "`Datum`" koja omogućava čuvanje datumskih podataka. Negdje u javnom dijelu klase (interfejsu) nalaze se sljedeće deklaracije:

```
enum Mjeseci {Januar = 1, Februar, Mart, April, Maj, Juni, Juli,
  August, Septembar, Oktobar, Novembar, Decembar};
enum Dani {Ponedjeljak = 1, Utorak, Srijeda, Cetvrtak, Petak, Subota,
  Nedjelja);
```

Pored ovoga, interfejs klase treba da sadrži sljedeće elemente:

a) Dva konstruktora sa tri parametra koji inicijaliziraju datum u skladu sa vrijednostima za dan, mjesec i godinu koji se zadaju putem parametara. Podržana su dva načina kako se može zadati mjesec: kao vrijednost tipa "`Mjeseci`" koji je lokalno definiran u interfejsu klase, ili kao cijeli broj u opsegu od 1–12 (zbog toga su i potrebna dva konstruktora). U slučaju da datum nije smislen, treba baciti izuzetak tipa "`domain_error`" uz prikladan prateći tekst.

b) Dvije metode nazvane "`Postavi`" koje u načelu obavljaju isti zadatak kao i konstruktori, ali olakšavaju naknadnu promjenu pohranjenih informacija.

c) Metode "`DajDan`", "`DajMjesec`" i "`DajGodinu`" koje služe za očitavanje informacija o danu, mjesecu i godini koje su pohranjene u datumu. Metode "`DajDan`" i "`DajGodinu`" daju kao rezultat cijeli broj, dok metoda "`DajMjesec`" daje rezultat tipa "`Mjeseci`".

d) Metodu "`DajImeMjeseca`" koja daje ime mjeseca koji se čuva u datumu. S obzirom da će rezultat ove funkcije uglavnom služiti samo za potrebe ispisivanja, rezultat ne treba biti tipa "`string`", nego tipa pokazivača na znakove (tako da se umjesto čitavog imena vraća se samo pokazivač na prvi znak imena).
e) Metodu "`DanUSedmici`" koja vraća dan u sedmici koji odgovara podacima zapamćenim u datumu (rezultat treba da bude tipa "`Dani`" koji je lokalno definiran u interfejsu klase), kao i metodu "`DajImeDanaUSedmici`" koja daje ime dana u sedmici koji odgovara datumu, na sličan način kao i metoda "`DajImeMjeseca`".
f) Preklopljene operatore "`++`" i "`--`", koji pomjeraju datum za jedan dan unaprijed odnosno unazad (potrebno je podržati i prefiksne i postfiksne verzije ovog operatora).
g) Preklopljene operatore "`+`" i "`-`", pri čemu je dozvoljeno je sabrati primjerak klase "`Datum`" sa pozitivnim cijelim brojem, odnosno oduzeti cijeli broj od primjerka klase "`Datum`", gdje se kao rezultat dobija novi primjerak klase "`Datum`" u kojem je datum pomjeren unaprijed odnosno unazad za broj dana iskazan drugim parametrom. Vodite računa da taj cijeli broj može biti i negativan.
h) Preklopljene operatore "`+=`" pri čemu izrazi "`d += n`" odnosno "`d –= n`" proizvode isto dejstvo kao i izrazi "`d = d + n`" odnosno "`d = d – n`".
i) Preklopljeni operator "`–`" koji omogućava oduzimanje dva primjera klase "`Datum`", pri čemu se kao rezultat dobija broj dana između dva datuma.
j) Preklopljene relacione operatore "`==`", "`!=`","`<`","`>`","`<=`" i "`>=`" koji daju rezultat poređenja dva datuma (u hronološkom poretku, tj. manji je onaj datum koji dolazi prije po kalendaru).
k) Preklopljeni operator "`<<`" za ispis datuma na ekran. Pri tome se prvo ispisuje redni broj dana, zatim tačka, zatim puno ime mjeseca (riječima) iza koje slijedi razmak, zatim redni broj godine iza kojeg takođe slijedi tačka i, konačno, ime odgovarajućeg dana u sedmici u zagradama. Na primjer, ukoliko je u datumu zapamćen datum 23. 5. 2012. poziv ove metode treba da na ekranu proizvede ispis "`23. Maj 2012. (Srijeda)`".
l) Preklopljeni operator "`<<`" za unos datuma sa tastature. Datum treba da se unosi u obliku *dan*/*mjesec*/*godina* (npr. 23/5/2012), pri čemu su *dan*, *mjesec* i *godina* cijeli brojevi. U slučaju da unos nije ispravan (što uključuje ne samo besmislene datume nego i slučajeve da nije unesena kosa crta ili da su unesena slova umjesto brojeva, itd.), ulazni tok treba postaviti u neispravno stanje.

Sve metode implementirajte izvan klase, osim trivijalnih metoda čija implementacija može stati u jedan red ekrana. Obavezno napišite i mali testni program u kojem će se testirati *svi zahtijevani elementi* ove klase.

Uputa: Što se tiče računanja dana u sedmici koji odgovara zadanom datumu, najbolje je da prvo izračunate broj dana koji je protekao između nekog po volji izabranog referentnog datuma za koji znate na koji je dan u sedmici pao i posmatranog datuma (što nije posve trivijalno uraditi na efikasan način, naročito zbog prestupnih godina), a zatim izračunate ostatak dijeljenja tog broja dana sa 7. Na osnovu tog ostatka i poznate činjenice na koji je dan pao referentni datum, vrlo lako se zaključuje na koji dan pada posmatrani datum.

3. Razvijte klasu "`TabelarnaFunkcija`" koja predstavlja tabelarno zadanu funkciju, tj. funkciju koja nije zadana analitički, nego kao skup parova realnih brojeva oblika (x_i, y_i) koji su dobijeni recimo mjerenjem. Ovi parovi će se čuvati u dinamički alociranom nizu objekata tipa "`Par`", pri čemu "`Par`" predstavlja običnu strukturu sa dva realna atributa "`x`" i "`y`". Interfejs ove klase treba sadržavati sljedeće elemente:

a) Konstruktor sa jednim parametrom koji vrši alokaciju prostora za čuvanje parova. Parametar predstavlja inicijalno predviđeni broj parova koji se mogu pohraniti (vidjećemo da se ovaj broj može naknadno mijenjati tokom rada klase). Ovaj parametar treba imati podrazumijevanu vrijednost 30, koja se koristi ukoliko korisnik ne navede parametar. U svakom slučaju, ovaj konstruktor se ne smije koristiti za automatsku konverziju cjelobrojnih podataka u tip ove klase.
b) Destruktor, koji oslobađa sve resurse koje su primjerci ove klase zauzeli tokom svog života.

c) Kopirajući konstruktor i kopirajući operator dodjele koji omogućavaju da se primjerci ove klase mogu bezbjedno kopirati i međusobno dodjeljivati na bazi dubokog kopiranja, te odgovarajući pomjerajući konstruktor i pomjerajući operator dodjele, koji optimiziraju ove operacije za slučaj kada su u igri privremeni objekti.

d) Metodu "`DodajPar`" sa dva parametra koja dodaje novi par (*x*, *y*) u postojeći skup parova. Parametri metode su upravo *x* i *y*. Ukoliko u skupu parova već postoji par čija je prva koordinata jednaka *x*, treba baciti izuzetak tipa "`logic_error`" uz odgovarajući prateći tekst (s obzirom da nije moguće imati dvije vrijednosti funkcije *y* za istu vrijednost *x*). Ukoliko se popuni sav alocirani prostor, treba alocirati novi prostor sa 20 dodatnih mjesta, iskopirati u njega sve pohranjene parove, obrisati stari (popunjeni) prostor i nastaviti dodavanje u novi prostor.

e) Sekvencijski konstruktor koji omogućava kreiranje objekata tipa "`TabelarnaFunkcija`" iz inicijalizacione liste čiji su elementi parovi brojeva unutar vitičastih zagrada. Recimo, trebala bi biti legalna konstrkucija poput "`TabelarnaFunkcija f{{1, 5}, {3, 4}, {4, 2}};`". U slučaju da postoje dva para sa istom prvom koordinatom, treba baciti izuzetak isto kao u metodi "`DodajPar`".

f) Metodu "`ObrisiSve`" koja briše sve unesene parove.

g) Metodu "`ObrisiPar`" sa jednim parametrom *x* koja briše iz kolekcije par čija je prva koordinata *x* ukoliko takav postoji, a u suprotnom baca izuzetak tipa "`domain_error`" uz odgovarajući prateći tekst.

h) Metodu "`TabelirajFunkciju`" sa četiri parametra *f*, *xmin*, *xmax* i *dx* pri čemu je *f* funkcija koja prima realni broj a vraća realni broj kao rezultat ili odgovarajući funkcijski objekat koji se može koristiti na isti način, dok su *xmin*, *xmax* i *dx* realni brojevi. Metoda treba da u kolekciju doda sve parove oblika (*x*, *f*(*x*)) za sve vrijednosti *x* od *xmin* do *xmax* u koraku *dx* (tj. da tabelira funkciju *f* na zadanom intervalu sa zadanim korakom i da pohrani rezultate tabeliranja u kolekciju).

i) Preklopljeni operator "`()`" koji vraća vrijednost funkcije u tački *x* (koja se zadaje kao argument) dobijenu postupkom linearne interpolacije (to je ono što se dobije kada se pretpostavi da su sve tačke (x_i,y_i) u rastućem poretku po *x*-ovima prosto spojene dužima). Formula za linearnu interpolaciju glasi $y = y' + (y'' - y')(x - x')/(x'' - x')$ gdje je x' najveća vrijednost među vrijednostima x_i koje su manje ili jednake od *x*, x'' je najmanja vrijednost među vrijednostima x_i koje su veće od *x*, dok su y' i y'' odgovarajuće vrijednosti y_i koje odgovaraju x' i x''. Ukoliko x' ili x'' ne postoje (npr. ako nema vrijednosti x_i koje su manje od *x*), interpolacija nije moguća i treba baciti izuzetak tipa "`domain_error`" uz odgovarajući prateći tekst.

Sve metode implementirajte izvan klase, osim trivijalnih metoda čija implementacija može stati u jedan red ekrana. Sve metode koje su po prirodi inspektori obavezno treba deklarirati kao takve. Obavezno napišite i mali testni program u kojem će se testirati *svi zahtijevani elementi* ove klase.

4. Meteorološke stanice moraju, između ostalog, vršiti čestu registraciju atmosferskog pritiska. Za tu svrhu, neka meteorološka stanica koristi program u kojem je definirana i implementirana klasa nazvana "`Pritisci`". Ova klasa omogućva čuvanje podataka o atmosferskim pritiscima za izvjesni vremenski period u vektoru realnih brojeva, kojem se pristupa preko odgovarajućeg privatnog atributa. Interfejs klase sadrzi sljedeće elemente:

   a) Konstruktor dva parametra koji predstavljaju minimalno dozvoljeni i maksimalno dozvoljeni atmosferski pritisak koji se može registrirati pri jednoj registraciji. Minimalno dozvoljeni pritisak mora biti manji od maksimalno dozvoljenog pritiska. U suprotnom, treba baciti izuzetak tipa "`range_error`" uz odgovarajući prateći tekst.

   b) Metodu "`RegistrirajPritisak`" koja vrši registraciju novog atmosferskog pritiska, koji se zadaje kao parametar. U slučaju da je zadani pritisak veći ili manji od maksimalno dozvoljenog pritiska (ove dvije vrijednosti su zadate prilikom konstrukcije objekta), treba baciti izuzetak tipa "`range_error`" uz odgovarajući prateći tekst.

   c) Metodu "`DajBrojRegistriranihPritisaka`" koja daje broj registriranih pritisaka.

   d) Metodu "`BrisiSve`" koja briše sve unesene pritiske.

e) Metode "`DajMinimalniPritisak`" i "`DajMaksimalniPritisak`" koje vraćaju kao rezultat minimalni i maksimalni registrirani pritisak (u slučaju da nema registriranih pritisaka, obje metode treba da bace izuzetak tipa "`range_error`" uz odgovarajući prateći tekst). Za realizaciju nije dozvoljeno koristiti petlje, nego isključivo odgovarajuće funkcije iz biblioteke "`algorithm`".

f) Metodu "`DajBrojDanaSaPritiscimaVecimOd`" koja vraća kao rezultat broj dana u kojima je pritisak bio veći od vrijednosti koja se zadaje kao parametar (u slučaju da nema registriranih pritisaka, metode treba da bace izuzetak tipa "`range_error`" uz odgovarajući prateći tekst). Za realizaciju 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`". Napomena: Ovdje ćete 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).

g) Metodu "`Ispisi`" koja ispisuje sve unesene (registrirane) pritiske sortirane u opadajućem poretku (tj. najveći pritisak se ispisuje prvi), pri čemu se svaki pritisak ispisuje u posebnom redu. Pri tome je neophodno koristiti funkcije i/ili funktore iz biblioteka "`algorithm`" i "`functional`" (upotreba lambda funkcija ili pmoćnih imenovanih funkcija kriterija nije dozvoljena). Ova funkcija obavezno treba biti inspektor funkcija, tj. treba biti deklarirana sa modifikatorom "**`const`**"!

h) Preklopljeni operator "`[]`" koji omogućava da se direktno pročita *i*-ti registrirani pritisak (numeracija ide od jedinice), pri čemu je indeks *i* parametar operatora. Ukoliko je indeks izvan dozvoljenog opsega, treba baciti izuzetak tipa "`range_error`" uz odgovarajući prateći tekst. Pri tome, ovaj operator se *ne može koristiti* za izmjenu podataka, odnosno ne može se koristiti sa lijeve strane znaka dodjele.

i) Preklopljeni operator "++" koji povećava sve registrirane pritiske za jedinicu (pri tome se za jedinicu povećava i informacija o minimalnom i maksimalnom dozvoljenom pritisku). Potrebno je podržati kako prefiksnu, tako i postfiksnu verziju ovog operatora. Za realizaciju nije dozvoljeno koristiti petlje, niti lambda funkcije ili pomoćne imenovane funkcije, nego isključivo odgovarajuće funkcije i/ili funktore iz biblioteka "`algorithm`" i "`functional`". Vrijedi ista napomena kao pod f).

j) Preklopljene operatore "+" i "−" koji djeluju na sljedeći način: Ukoliko je "`X`" objekat tipa "`Pritisci`", a "`Y`" realni, tada je "`X + Y`" novi objekat tipa "`Pritisci`" u kojem su svi registrirani pritisci povećani za iznos "`Y`" (u novodobijenom objektu treba ažurirati informaciju o minimalnom i maksimalnom dozvoljenom pritisku). Izraz "`Y + X`" treba da ima isto značenje kao i izraz "`X + Y`". Izraz "`X - Y`" u slučaju da je "`X`" objekat tipa "`Pritisci`", a "`Y`" realan broj interpretira se analogno, dok je tada "`Y - X`" objekat tipa "`Pritisci`" u kojem su svi registrirani pritisci oduzeti od vrijednosti "`Y`". U slučaju kada su i "`X`" i "`Y`" objekti tipa "`Pritisci`", tada je izraz "`X - Y`" novi objekat tipa "`Pritisci`" koji sadrži *razlike* odgovarajućih količina padavina iz objekata "`X`" i "`Y`" (informacija o maksimalnom dozvoljenom pritisku u tako kreiranom objektu jednaka je razlici maksimalnog dozvoljenog pritiska iz objekta "`X`" i minimalnog dozvoljenog pritiska iz objekta "`Y`", dok je informacija o minimalnom dozvoljenom pritisku jednaka razlici minimalnog dozvoljenog pritiska iz objekta "`X`" i maksimalnog dozvoljenog pritiska iz objekta "`Y`"). Pri tome se podrazumijeva da "`X`" i "`Y`" sadrže isti broj registriranih količina padavina (u suprotnom, treba baciti izuzetak tipa "`range_error`" uz odgovarajući prateći tekst). U novokreiranom objektu informaciju o minimalnom i maksimalnom pritisku treba ažurirati u skladu sa minimalnim i maksimalnim elementom pohranjenim u novokreiranom objektu. U svim ostalim slučajevima, značenje izraza "`X + Y`" odnosno "`X - Y`" nije definirano. Za realizaciju ovih operatora nije dozvoljeno koristiti petlje, niti lambda funkcije ili pomoćne imenovane funkcije, nego isključivo odgovarajuće funkcije i/ili funktore iz biblioteka "`algorithm`" i "`functional`". Vrijedi ista napomena kao pod f).

k) 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.

l) Preklopljeni unarni operator "-" koji daje kao rezultat novi objekat tipa "`Pritisci`" u kojem su svi pritisci oduzeti od maksimalno dozvoljenog pritiska. Informaciju o minimalno dozvoljenom pritisku u novokreiranom objektu treba postaviti na nulu, a informaciju o maksimalno dozvoljenom pritisku na iznos razlike između maksimalno i minimalno dozvoljenog pritiska u izvornom objektu. Za realizaciju nije dozvoljeno koristiti petlje, niti lambda funkcije ili pomoćne imenovane funkcije, nego isključivo odgovarajuće funkcije i/ili funktore iz biblioteka "`algorithm`" i "`functional`". Vrijedi ista napomena kao pod f).

m) Preklopljene relacione operatore "`==`" i "`!=`" koje ispituju da li su dva objekta tipa "`Padavine`" jednaka ili nisu. Dva objekta ovog tipa smatraju se jednakim ukoliko sadrže isti broj registriranih količina padavina, i ukoliko su sve odgovarajuće registrirane količine padavina oba objekta jednake. Za realizaciju ovih operatora nije dozvoljeno koristiti petlje, nego isključuvo odgovarajuće funkcije i/ili funktore iz biblioteka "`algorithm`" i "`functional`".

Implementirajte klasu sa navedenim svojstvima. Sve neophodne atribute treba obavezno izvesti kao privatne članove klase, a sve metode implementirajte izvan klase, osim metoda čija je implementacija dovoljno kratka, u smislu da zahtijeva recimo jednu ili dvije naredbe. Sve 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.