Zadaća 5.

Ova zadaća nosi ukupno 3 poena, pri čemu prvi zadatak nosi 1,3 poena, drugi zadatak nosi 0,7 poena, dok treći zadatak nosi 1 poen. Sva tri zadatka se mogu uraditi na osnovu gradiva sa prvih 11 predavanja i pretpostavljenog predznanja iz predmeta "Osnove računarstva". Rok za predaju ove zadaće je četvrtak, 7. VI 2012. (do kraja dana) i ne može se produžiti. Zadaće se predaju putem Zamgera.

1. 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.
   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 "`DajBrojDanaUMjesecu`" koja vraća koliko ukupno dana ima onaj mjesec koji je trenutnu pohranjen u datumu.
   f) Metodu "`DaLiJePrestupna`" vraća informaciju o tome da li je godina koja se čuva u datumu prestupna ili ne.
   g) Metode "`DajDanUGodini`" i "`DajSedmicu`" vraćaju kao rezultat redni broj dana odnosno sedmice koji odgovaraju posmatranom datumu unutar godine koja se čuva u datumu.
   h) 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`".
   i) Preklopljene operatore "`++`" i "`--`", koji pomjeraju datum za jedan dan unaprijed odnosno unazad (potrebno je podržati i prefiksne i postfiksne verzije ovog operatora).
   j) 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.
   k) Preklopljene operatore "`+=`" pri čemu izrazi "`d += n`" odnosno "`d –= n`" proizvode isto dejstvo kao i izrazi "`d = d + n`" odnosno "`d = d – n`".
   l) Preklopljeni operator "`–`" koji omogućava oduzimanje dva primjera klase "`Datum`", pri čemu se kao rezultat dobija broj dana između dva datuma.
   m) Preklopljene relacione operatore "`==`", "`!=`","`<`","`>`","`<=`" i "`>=`" koji daju rezultat poređenja dva datuma (u hronološkom poretku, tj. manji je onaj datum koji dolazi prije po kalendaru).

n) 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)`".

o) 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 baš 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.

2. Linearne funkcije jedne realne promjenljive predstavljaju vjerovatno najviše korištene funkcije u svim oblastima nauke i tehnike. To su funkcije koje se mogu predstaviti analitičkim izrazom oblika $f(x) = kx + l$ gdje je x realna promjenljiva, a k i l su realni parametri nazvane redom *koeficijent pravca* i *slobodni član*. Stoga je od koristi razviti opću klasu (možemo je nazvati recimo "`LinFun`") pomoću koje se mogu kreirati objekti koji se ponašaju poput linearnih funkcija sa parametrima koji se mogu zadavati prilikom kreiranja objekta (i eventualno naknadno mijenjati). Na primjer, takvi objekti mogu se koristiti kao u sljedećem primjeru:

```
LinFun f(3, 5);  // Kreira lin. funkciju nazvanu "f", f(x) = 3 x + 5
...
cout << f(2);   // Ispisuje vrijednost funkcije "f" za argument 2 (11)
```

Pored toga, predviđeno je da se sa objektima tipa "`LinFun`" mogu vršiti i različite manipulacije, u skladu sa opisom koji slijedi. Klasa "`LinFun`" kao jedine svoje atribute sadrži koeficijent pravca i slobodni član odgovarajuće linearne funkcije. Definirajte i implementirajte klasu sa navedenim osobinama, znajući da njen interfejs sadrži sljedeće elemente (sve metode koje su inspektori trebaju obavezno biti označene kao takve):

a) Konstruktore, koji omogućavaju kreiranje objekata tipa "`LinFun`". Predviđeno je da se objekti ovog tipa mogu kreirati na više različitih načina. Moguće je pri kreiranju zadati dva parametra, koji redom predstavljaju koeficijent pravca i slobodni član linearne funkcije koja se kreira. Dalje, moguće je zadati samo jedan parametar, koji tada predstavlja slobodni član, dok se koeficijent pravca postavlja na nulu (time se faktički kreira konstantna linearna funkcija). Konačno, objekte tipa "`LinFun`" moguće je kreirati i bez zadavanja ikakvih parametara. U tom slučaju, i koeficijent pravca i slobodni član se postavljaju na nulu. Potrebno je podržati automatsku pretvorbu po kojoj se realni brojevi automatski mogu tretirati i kao konstantne linearne funkcije (tj. one sa koeficijentom pravca jednakim nuli). Napomena: mada se objekti tipa "`LinFun`" mogu kreirati na tri različita načina, to ne znači nužno da moraju postojati tri konstruktora – ako možete proći sa manje, tim bolje po Vas.

b) Metodu "`Postavi`" sa dva parametra koja omogućava naknadnu promjenu koeficijenta pravca i slobodnog člana već definirane linearne funkcije (tj. objekta tipa "`LinFun`").

c) Pristupne metode "`DajKoeficijentPravca`" i "`DajSlobodniClan`" bez parametara koje omogućavaju da se saznaju vrijednosti koeficijenta pravca odnosno slobodnog člana.

d) Preklopljeni unarni operator "`-`" koji daje kao rezultat negiranu linearnu funkciju, tj. funkciju čiji su koeficijent pravca i slobodni član negirani u odnosu na izvornu funkciju.

e) Preklopljene binarne operatore “+” i “–” koji omogućavaju da se saberu odnosno oduzmu dvije linearne funkcije, pri čemu se kao rezultati dobijaju nove linearne funkcije (njihovi koeficijenti pravca odnosno slobodni članovi jednaki su zbiru odnosno razlici koeficijenata pravca i slobodnih članova oba sabirka). Napomena: radi automatske konverzije brojeva u linearne funkcije, ovo će automatski omogućiti i sabiranje odnosno oduzimanje funkcije i broja, odnosno broja i funkcije.

f) Preklopljeni binarni operator “`*`” koji omogućava množenje linearne funkcije realnim brojem, odnosno realnog broja linearnom funkcijom. Kao rezultat se dobija nova linearna funkcija čiji su koeficijent pravca i slobodni član pomnoženi zadanim brojem. Množenje dvije linearne funkcije ne treba podržati, jer rezultat takvog množenja nije linearna funkcija.

g) Preklopljeni binarni operator “`/`” koji omogućava dijeljenje linearne funkcije realnim brojem. Kao rezultat se dobija nova linearna funkcija čiji su koeficijent pravca i slobodni član podijeljeni zadanim brojem. Dijeljenje dvije linearne funkcije kao i dijeljenje realnog broja linearnom funkcijom ne treba podržati, jer rezultat takvog dijeljenja nije linearna funkcija.

h) Preklopljene operatore “`+=`”, “`-=`”, “`*=`” i “`/=`” koji omogućavaju da izrazi poput “`X += Y`”, “`X -= Y`”, “`X *= Y`” i “`X /= Y`” imaju isto značenje kao i izrazi “`X = X + Y`”, “`X = X - Y`”, “`X = X * Y`” i “`X = X / Y`” kad god ovi imaju smisla.

i) Preklopljene operatoe “`++`” i “`--`” koji povećavaju odnosno smanjuju slobodni član u linearnoj funkciji na koju je primijenjen za jedinicu. Potrebno je podržati i prefiksne i postfiksne verziju ovog operatora.

j) Preklopljeni operator “`()`” koji omogućava izračunavanje vrijednosti linearne funkcije za vrijednost argumenta koja se zadaje kao parametar.

k) Metodu “`DajInverznu`” koja daje kao rezultat novu linearnu funkciju koja je inverzna funkcija funkcije nad kojom je metoda primijenjena. Napomena: ako neka funkcija ima koeficijent pravca k i slobodni član l, njena inverzna funkcija ima koeficijent pravca $1/k$ i slobodni član $-l/k$. U slučaju dijeljenja nulom, treba baciti izuzetak.

Obavezno napišite i mali testni program u kojem ćete testirati sve elemente napisane klase.

3. Za potrebe neke meteorološke stanice neophodno je vrsiti čestu registraciju količine padavina. Za tu svrhu meteorološka stanica koristi računarski program u kojem je definirana i implementirana klasa nazvana “`Padavine`”. Ova klasa omogućava čuvanje podataka o količini padavina za izvjesni vremenski period u dinamički alociranom nizu cijelih brojeva, kojem se pristupa preko odgovarajućeg privatnog atributa (količina padavina se zadaje u centimetrima, zaokruženo na cijeli broj). Pored ovog atributa (i eventualno drugih privatnih atributa neophodnih za normalno funkcioniranje klase), poznato je da klasa sadrži sljedeće elemente:

a) Konstruktor sa dva parametra, koji redom predstavljaju maksimalan broj količina padavina koje se mogu registrirati, odnosno maksimalno dozvoljenu količinu padavina. Ovaj konstruktor je odgovoran za dinamičku alokaciju memorije.

b) Destruktor, koji oslobadja resurse koji su zauzeti od strane ove klase.

c) Konstruktor kopije, koji obezbjedjuje siguran prenos objekata tipa “`Padavine`” kao parametara po vrijednosti u funkcije i njihovo vraćanje kao rezultata iz funkcije, kao i preoklopljeni operator dodjele “=” koji obezbjeđuje sigurnu dodjelu jednog objekta tipa “`Padavine`” drugom.

d) Metodu koja vrši registraciju nove količine padavina, pri čemu se količina padavina koja se registrira prenosi kao parametar metode. U slučaju da se dostigne maksimalan broj količina padavina koje se mogu registrirati, ili u slučaju da je količina padavina manja od nule ili veća od maksimalne dozvoljene količine padavina, metoda treba da baci izuzetak.

e) Metodu koja daje broj registriranih količina padavina.

f) Metodu koja briše sve unesene količine padavina.

g) Metode koje vraćaju kao rezultat minimalnu i maksimalnu količinu padavina (u slučaju da nema registriranih količina padavina, obje metode treba da bace izuzetak). Za

realizaciju nije dozvoljeno koristiti petlje, nego odgovarajuće funkcije i/ili funktore iz biblioteka “`algorithm`” i “`functional`”.

h) Metode koje vraćaju kao rezultat broj dana u kojima je količina padavina bila veća odnosno manja od od vrijednosti koja se zadaje kao parametar (u slučaju da nema registriranih količina padavina, metode treba da bace izuzetak). Za realizaciju nije dozvoljeno koristiti petlje, nego odgovarajuće funkcije i/ili funktore iz biblioteka “`algorithm`” i “`functional`”.

i) Metodu koja ispisuje sve unesene (registrirane) količine padavina sortirane u opadajućem poretku (tj. najveća količina padavina se ispisuje prva), pri čemu se svaka količina padavina ispisuje u posebnom redu. Pri tome je neophodno koristiti funkcije i/ili funktore iz biblioteka “`algorithm`” i “`functional`”.

j) Preklopljeni operator “`[]`” koji omogućava da se direktno pristupi *i*-toj registriranoj količini padavina (numeracija ide od jedinice). Ukoliko je indeks izvan dozvoljenog opsega, treba baciti izuzetak.

k) Preklopljeni operator “`++`” koji povećava sve registrirane količine padavina za jedinicu (pri tome se za jedinicu povećava i informacija o maksimalno dozvoljenoj količini padavina). Potrebno je podržati kako prefiksnu, tako i postfiksnu verziju ovog operatora. Za realizaciju nije dozvoljeno koristiti petlje, nego odgovarajuće funkcije i/ili funktore iz biblioteka “`algorithm`” i “`functional`”.

l) Preklopljene operatore “`+`” i “`-`” koji djeluju na sljedeći način: Ukoliko je “`X`” objekat tipa “`Padavine`”, a “`Y`” cijeli broj, tada je “`X + Y`” novi objekat tipa “`Padavine`” u kojem su sve registrirane količine padavina povećane za iznos “`Y`” (u novodobijenom objektu treba ažurirati informaciju o maksimalno dozvoljenoj količini padavina). Na sličan način se interpretira i izraz “`X - Y`” u slučaju da je “`X`” objekat tipa “`Padavine`”, a “`Y`” cijeli broj. Pri tome, ukoliko se kao rezultat sabiranja ili oduzimanja dobije da neka od količina padavina postane negativna, treba baciti izuzetak. U slučaju kada su i “`X`” i “`Y`” objekti tipa “`Padavine`”, tada je izraz “`X - Y`” novi objekat tipa “`Padavine`” koji sadrži *razlike* odgovarajućih količina padavina iz objekata “`X`” i “`Y`”. U ovom posljednjem slučaju se podrazumijeva da “`X`” i “`Y`” sadrže isti broj registriranih količina padavina, kao i da su registrirane padavine u objektu “`X`” veće ili jednake od odgovarajućih registriranih padavina u objektu “`Y`” (u suprotnom, treba baciti izuzetak). U svim ostalim slučajevima, značenje izraza “`X + Y`” odnosno “`X - Y`” nije definirano. Za realizaciju ovih operatora nije dozvoljeno koristiti petlje, nego odgovarajuće funkcije i/ili funktore iz biblioteka “`algorithm`” i “`functional`”.

m) 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.

n) Preklopljeni unarni operator “`-`” koji daje kao rezultat novi objekat tipa “`Padavine`” u kojem su sve količine padavina oduzete od maksimalno dozvoljene količine padavina. Za realizaciju nije dozvoljeno koristiti petlje, nego odgovarajuće funkcije i/ili funktore iz biblioteka “`algorithm`” i “`functional`”.

o) 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 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. Obavezno napišite i mali testni program u kojem će se testirati sve elemente napisane klase.