Zadaća 5.

Ova zadaća nosi ukupno 5 poena, pri čemu prvi zadatak nosi 1,2 poena, drugi zadatak 1,4 poena, treći zadatak 0,8 poena, četvrti zadatak 0,4 poena i peti zadatak 1,2 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 četvrtak, 2. VI 2016. do 5.00.

NAPOMENA: U slučaju da smatrate da Vam trebaju neke pomoćne funkcije za realizaciju neophodnih funkcionalnosti, u zadacima, možete ih slobodno dodati u privatni dio klase. Međutim, interfejs klase *ne smijete mijenjati*, osim ukoliko neki detalj iz postavke zadatka izričito i nedvosmisleno ne ukazuje na to da treba praviti izmjene u interfejsu klase.

1. 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 (također cijeli broj), ime i prezime studenta, adresu studenta, te broj telefona studenta (ova posljednja tri parametra su po tipu konstantni nizovi znakova). Broj indeksa studenta mora biti peterocifreni broj, a godina studija broj u opsegu od 1 do 8 (4 i 5 odgovara master studiju, a 6, 7 i 8 doktorskom studiju). Ime i prezime te adresa mogu biti bilo kakvi, ali 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`" kojima se mogu pročitati informacije o studentu (posljednje 3 metode vraćaju rezultat tipa "`string`"), te metodu "`Ispisi`" koja ispisuje podatke o studentu na ekran. Format ispisa treba da bude kao u sljedećem primjeru (iza dvotačke je tačno jedan razmak):

```
Broj indeksa: 12345
Godina studija: 3/I
Ime i prezime: Dino Dinic
Adresa: Trg vjecnih studenata 12
Telefon: 074/239-175
```

   Godina studija se ispisuje u formatu *godina*/*ciklus* gdje se ciklus ispisuje kao rimski broj, npr. 2/I za drugu godinu dodiplomskog studija, 1/II za prvu godinu master studija, te 3/III za treću godinu doktorskog studija.

   Klasa "`Laptop`" posjeduje konstruktor sa 3 parametra. Prvi parametar je cijeli broj koji predstavlja evidencijski broj laptopa, dok su drugi i treći parametar konstantni nizovi znakova koji redom predstavljaju naziv laptopa, te dodatne napomene o osnovnim karakteristikama laptopa. 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`" koje redom vraćaju evidencijski broj, naziv i napomene o karakteristikama laptopa (posljednje dvije metode vraćaju rezultat tipa "`string`"), kao i metode "`Zaduzi`", "`Razduzi`", "`DaLiJeZaduzen`" 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 "`KodKogaJe`" (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). Konačno, metoda "`Ispisi`" 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. Podaci o svakom studentu odnosno laptopu čuvaju se u dinamički alociranim varijablama, kojima se pristupa pomoću dva vektora čiji su elementi pametni pokazivači na studente (tj. na objekte tipa "`Student`") odnosno pametni pokazivači na laptope (tj. na objekte tipa "`Laptop`"). Objekti tipa "`Administracija`" ne zahtijevaju nikakve parametre za kreiranje. Njihovo kopiranje i međusobno dodjeljivanje treba zabraniti, ali "pomjeranje" treba biti podržano (odnosno, treba biti podržan mogućnost da se jedan objekat tipa "`Administracija`" uz pomoć "`move`" funkcije prebaci u drugi). Što se tiče interfejsa klase, on treba da sadrži sljedeće metode. 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". Metoda "`RegistrirajNoviLaptop`" radi istu stvar, 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 "Dino Dinic (1234)"). 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 Dino Dinic (1234) 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.

2. Dizajnirajte i implementirajte klasu "`VelikiBroj`" koja omogućava rad sa cijelim brojevima proizvoljne veličine (tačnije čija je veličina ograničena samo količinom raspoložive memorije). Primjerci ove klase (tj. promjenljive tipa "`VelikiBroj`") trebali bi da imaju sva svojstva koja imaju i obične cjelobrojne promjenljive. Drugim riječima, na objekte tipa "`VelikiBroj`" moraju se moći primjenjicati standardni binarni operatori "`+`", "`-`", "`*`", "`/`" i "`%`", operator dodjele "`=`", kombinirani operatori "`+=`", "`-=`", "`*=`", "`/=`" i "`%=`", unarni operatori "`+`", "`-`", "`++`" i "`--`" (pri čemu ova posljednja dva operatora trebaju biti podržani i u prefiksnoj i u postfiksnoj verziji), kao i

relacioni operatori "`==`", "`!=`", "`<`", "`>`", "`<=`" i "`>=`". Također, objekti tipa "`VelikiBroj`" moraju se moći ispisivati na izlazni tok (recimo na ekran) pomoću operatora "`<<`" odnosno čitati iz ulaznog toka (recimo sa tastature) pomoću operatora "`>>`". U slučaju dijeljenja nulom (u operatorima "`/`", "`%`","`/=`" i "`%=`") treba baciti izuzetak tipa "`overflow_error`" uz prateći tekst "Dijeljenje nulom".

Inicijalizacija objekata tipa "`VelikiBroj`", u slučaju da vrijednost koju želimo da stavimo u njega ne prelazi opseg klasičnog cjelobrojnog tipa (tj. tipa "**`int`**") treba da se može izvoditi na identičan način kao inicijalizacija običnih cjelobrojnih promjenljivih. Drugim riječima, inicijalizacije poput sljedećih treba da budu dozvoljene (sve tri su ekvivalentne):

```
VelikiBroj a = 12345;
VelikiBroj a(12345);
VelikiBroj a{12345};
```

Također, inicijalizacija treba da se može vršiti navođenjem njegovih cifara unutar niza znakova između dva navodnika, što se mora koristiti ukoliko želimo zadati broj čija je vrijednost prevelika za tip "**`int`**". Na primjer, takve su inicijalizacije pomoću sljedećih:

```
VelikiBroj b = "-23465672345671234765723445612";
VelikiBroj b("-23465672345671234765723445612");
VelikiBroj b{"-23465672345671234765723445612"};
```

Između znakova navoda trebale bi se nalaziti isključivo cifre, pri čemu se eventualno ispred prve cifre može nalaziti predznak "+" ili "−" (prazan string, ili string koji sadrži samo predznak, interpretira se kao 0). Ukoliko to nije ispunjeno (tj. ukoliko broj nije sintaksno ispravan), treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Neispravan broj". Sama deklaracija oblika

```
VelikiBroj c;
```

treba kreirati objekat inicijaliziran na nulu. Treba biti podržana automatska pretvorba običnih cjelobrojnih vrijednosti i nizova cifara između navodnika u objekte tipa "`VelikiBroj`", tako da se u izrazima mogu miješati objekti tipa "`VelikiBroj`" i obični cijeli brojevi, odnosno nizovi znakova između navodnika koji sadrže sintaksno ispravan broj. Na primjer, sljedeći isječak

```
VelikiBroj br = "1234567890987654321";
std::cout << br + 999 << std::endl;
std::cout << "1324354657687980" + br << std::endl;
```

treba da na ekran ispiše "1234567890987655320" i "1235892245645342301". Naravno, mora biti moguće naknadno *dodijeliti* objektu tipa "`VelikiBroj`" običan broj ili niz cifara između navodnika, a ne samo *inicijalizirati* objekat prilikom njegovog kreiranja.

Pored toga, kad god je to moguće, potrebno je omogućiti pretvorbu objekata tipa "`VelikiBroj`" u sve cjelobrojne tipove (konkretno tipove "**`char`**", "**`short`**", "**`int`**", "**`long int`**" i "**`long long int`**", kao i njihove nepredznačne varijante sa modifikatorom "**`unsigned`**"), pa čak i u tip "**`bool`**" (pri čemu konverzija u tip "**`bool`**" daje "**`true`**" ako i samo ako je broj različit od nule, inače daje "**`false`**"). Navedenu pretvorbu ćete omogućiti implementiranjem operatora za pretvorbu, za svaki od navedenih tipova posebno (naravno, moguće je unutar jedne izvedbe pozivati neku drugu izvedbu da se skrati kôd). Ukoliko broj pohranjen u objektu tipa "`VelikiBroj`" ne može da stane u željeni tip, konverzija nije moguća i treba baciti izuzetak tipa "`range_error`" uz prateći tekst "Konverzija nije moguca". Na primjer, ukoliko je "`Fun`" neka funkcija koja ima parametar tipa "**`short int`**", navedeni isječak treba da radi u skladu sa očekivanjima:

```
VelikiBroj b("1234");
Fun(static_cast<short>(b));
```

Pri tome, treba *onemogućiti automatsku konverziju* objekata tipa "`VelikiBroj`" u cjelobrojne tipove (tj. zabraniti da se konverzija vrši ukoliko je nismo eksplicitno zahtijevali), tako da u prethodnom primjeru direktan poziv "`Fun(b)`" ne treba da radi. Podsjetimo se da se ključna riječ "**`int`**" može izostaviti kod tipova "**`short int`**", "**`long int`**" i "**`long long int`**", tako da je recimo "**`short`**" isto što i "**`short int`**". Također, samo "**`unsigned`**" je isto što i "**`unsigned int`**". Vi možete koristiti ono šta Vam više odgovara.

Klasu ćete realizirati tako što ćete cifre velikog broja čuvati vektoru čiji su elementi tipa "**`int`**", svaka cifra u posebnom elementu vektora, nakon čega se, računske operacije se mogu izvesti klasičnim postupkom kako se računske operacije sa cijelim brojevima izvode na papiru, cifru po

cifru. Doduše, sa aspekta utroška memorije bilo bi mnogo racionalnije koristiti string ili vektor čiji su elementi tipa "`char`", ili čuvati više od jedne cifre u jednom elementu vektora, ali ovdje se traži da implementirate baš takvo "školsko" rješenje. Svakako će utrošak memorije postati značajan tek pri toliko velikim brojevima za koje "školski" postupak izvođenja računskih operacija svakako traje predugo da bi klasa uopće bila upotrebljiva. Još jedna dobra ideja je da se u elementima vektora ne čuvaju individualne cifre, već skupine cifara (na taj način se smanjuje broj neophodnih računskih operacija, ali se komplicira izvedba prenosa), ali Vi nećete izvoditi te optimizacije, nego ćete striktno postupiti kako je rečeno: jedna cifra u jedan element niza.

Izvedbe svih binarnih operatora koji ne mijenjaju svoje argumente trebaju biti realizirane kao obične funkcije (nečlanice), dok izvedbe svih binarnih operatora koji mijenjaju svoj prvi argument kao i izvedbe unarnih operatora trebaju biti realizirane kao funkcije članice. Pri tome parovi srodnih operatora poput "+" i "+=" *ne smiju biti realizirani jedan preko drugog* (tj. niti "+" smije biti realiziran preko "+=", niti obrnuto).

Napisanu klasu iskoristite u testnom program treba da ilustrira ispravnost rada klase na velikom broju različitih testnih primjera, koji uključuju raznovrsne situacije. Na primjer, svaku od binarnih operacija treba testirati sa različitim kombinacijama predznaka operanada. Dalje, treba testirati kako situacije kada oba operanda imaju jednak broj cifara, tako i situacije kada operandi imaju različit broj cifara. Drugim riječima, testni program treba da sa što je god moguće većom sigurnošću potvrdi da klasa *zaista radi*. Konačno, testni program kao demonstraciju treba da izračuna faktorijel od 1000, pomoću sekvence poput sljedeće:

```
VelikiBroj fakt = 1;
for(int i = 1; i <= 1000; i++) fakt *= i;
std::cout << fakt;
```

Tačan rezultat trebao bi da bude:

```
402387260077093773543702433923003985719374864210714632543799910429938512398629020
592044208486969404800479988610197196058631666872994808558901323829669944590997424
504087073759918823627727188732519779505950995276120874975462497043601418278094646
496291056393887437886487337119181045825783647849977012476632889835955735432513185
232395846307555740911426241747434934755342864657661166779739666882029120737914385
371958824980812686783837455973174613608537953452422158659320192809087829730843139
284440328123155861103697680135730421616874760967587134831202547858932076716913244
842623613141250878020800026168315102734182797770478463586817016436502415369139828
126481021309276124489635992870511496497541990934222156683257208082133318611681155
361583654698404670897560290095053761647584772842188967964624494516076535340819890
138544248798495995331910172335555660213945039973628075013783761530712776192684903
435262520001588853514733161170210396817592151090778801939317811419454525722386554
146106289218796022383897147608850627686296714667469756291123408243920816015378088
989396451826324367161676217916890977991190375403127462228998800519544441428201218
736174599264295658174662830295557029902432415318161721046583203678690726015878352
075151628422554026517048330422614397428693306169089796848259012545832716822645806
652676995865268227280707578139185817888965220816434834482599326604336766017699961
283186078838615027946595513115655203609398818061213855860030143569452722420634463
179746059468257310379008402443243846565724501440282188525247093519062092902313649
327349756551395872055965422874977401141334696271542284586237738753823048386568897
646192738381490014076731044664025989949022222176590433990188601856652648506179970
235619389701786004081188972991831102117122984590164192106888438712185564612496079
872290851929681937238864261483965738229112312502418664935314397013742853192664987
533721894069428143411852015801412334482801505139969429015348307764456909907315243
327828826986460278986432113908350621709500259738986355427719674282224875758676575
234422020757363056949882508796892816275384886339690995982628095612145099487170124
451646126037902930912088908694202851064018215439945715680594187274899809425474217
358240106367740459574178516082923013535808184009699637252423056085590370062427124
341690900415369010593398383577793941097002775347200000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000000000000000000000000
00000000
```

Napomena 1: Od studenata se ne očekuje da implementacija ove klase bude pretjerano efikasna (pri testiranju neće se testirati efikasnost). Ipak, treba da se zna da ukoliko Vam na današnjim računarima vrijeme izvršavanja prethodno navedenog primjera koji računa faktorijel od 1000 traje duže od jedne sekunde, u pitanju je *vrlo loša implementacija*.

Napomena 2: Nepoštovanje izričitih naredbi i ograničenja koja su navedena u postavci (poput drugačijeg načina čuvanja cifara broja, nepropisnih izvedbi operatora, itd.) biće kažnjeno davanjem 0 poena na zadatak, bez obzira na autotestove.

3. Jedna od najkorisnijih stvari koje su ikada uvedene u matematici su Fourierovi redovi, koje je, tek da se zna, uveo inženjer (a ne matematičar) J. Fourier početkom 19. vijeka sa ciljem rješavanja nekih praktičnih problema termodinamike (problem distribucije toplote u čvrstim tijelima). Kasnije su pomoću Fourierovih redova riješeni mnogi dotada neriješeni problemi fizike i tehnike, prvenstveno problemi iz elektrostatike, teorije oscilacija i prostiranja valova, itd. Značaj ovih redova je toliko veliki da bez Fourierovih redova danas ne bi bilo mobilne telefonije, CD playera, JPG kompresije slike, MP3 kompresije zvučnih podataka, i mnogih drugih stvari bez kojih je život danas nezamisliv. Stoga je Vaš zadatak da dizajnirate i implementirate jednu klasu u čast ovom genijalnom izumu koja će se, naravno, zvati "`FourierovRed`". Fourierovi (ili trigonometrijski) redovi su redovi oblika

$$f(x) = \frac{a_0}{2} + \sum_{k=1}^{\infty} a_k \cos\frac{2k\pi x}{\mathrm{T}} + b_k \sin\frac{2k\pi x}{\mathrm{T}}$$

gdje su T, a_k, $k = 0 \,..\, \infty$ i b_k, $k = 1 \,..\, \infty$ neke realne konstante. Ukoliko ovaj red konvergira, on definira neku periodičnu funkciju sa periodom T. Naravno, u računarskim implementacijama sumiranje nikada ne može ići do u beskonačnost, tako da ćete Vi zapravo implementirati "sasječenu" verziju Fourierovog reda kod koje sumiranje ide do neke gornje granice N. Takvi "podrezani" Fourierovi redovi nazivaju se Fourierovi ili trigonometrijski polinomi, pri čemu se broj N naziva stepen Fourierovog polinoma (u skladu sa tim. prikladniji naziv za klasu bi bio FourierovPolinom, ali FourierovRed zvuči više kûl). Samim tim, pitanje konvergencije otpada, jer će takvi redovi imati samo konačno mnogo članova. Za dovoljno veliko N, Fourierovi polinomi će predstavljati sasvim dobru aproksimaciju pravih Fourierovih redova (ukoliko oni konvergiraju). Interfejs klase "`FourierovRed`" treba da sadrži sljedeće elemente:

- Konstruktor čiji su parametri realan broj T i dvije inicijalizacione liste, od kojih prva sadrži koeficijente a_0, a_1, a_2 itd. a druga koeficijente b_1, b_2, b_3 itd. Stepen N se automatski određuje na osnovu broja elemenata u ovim listama. U slučaju da dužine ove dvije liste nisu međusobno saglasne, za nedostajuće koeficijente se uzima da su nule. Na primjer,ukoliko je u prvoj listi posljednji koeficijent a_4, a u drugoj b_6, uzima se da su koeficijenti a_5 i a_6 nule.
- Konstruktor čiji su parametri pozitivni cijeli broj N (stepen), realan broj T, te dvije funkcije koje primaju cijeli broj, a vraćaju realan broj kao rezultat (ili bilo šta što se ponaša kao funkcija koja može primiti cijeli broj kao parametar, a vraća nešto što se može interpretirati kao realan broj). Ova posljednja dva parametra predstavljaju funkcije kojima se definiraju izrazi za a_k i b_k u funkciji od k. Na primjer, ukoliko želimo da definiramo Fourierov polinom "`f`" desetog stepena, sa periodom 2π, a čiji su koeficijenti zadani formulama $a_0 = 1$, $a_k = 1/k$ za $k \in \mathbb{N}$ i $b_k = 0$ za $k \in \mathbb{N}$, to možemo uraditi ovako:

```
FourierovRed f(10, 2 * PI,
  [](int k) { return k == 0 ? 1 : 1/k},
  [](int k) { return 0; }
);
```

  U slučaju da N nije pozitivan, baca se izuzetak tipa "`domain_error`" uz prateći tekst "Stepen mora biti pozitivan"
- Konstruktor čiji su parametri pozitivni cijeli broj N (stepen), realni brojevi p i q, funkcija f koja prima realan broj i vraća također realan broj kao rezultat (ili bilo šta što se ponaša kao funkcija koja može primiti realni broj kao parametar i vraća nešto što se može interpretirati kao realan broj), te pozitivni cijeli broj M čija će uloga biti uskoro objašnjena. Ovaj konstruktor kreira Fourierov polinom N-tog stepena koji aproksimira funkciju f na intervalu (p, q). Naime, pokazuje se da ukoliko se koeficijenti a_k, $k \in \mathbb{N} \cup \{0\}$ i b_k, $k \in \mathbb{N}$ izračunaju po formulama

$$a_k = \frac{2}{\mathrm{T}} \int_p^q f(x) \cos\frac{2k\pi x}{\mathrm{T}}\,\mathrm{d}x, \qquad b_k = \frac{2}{\mathrm{T}} \int_p^q f(x) \sin\frac{2k\pi x}{\mathrm{T}}\,\mathrm{d}x$$

gdje je $\mathrm{T} = q - p$, tada pod izvjesnim,vrlo blagim uvjetima (poznatim kao Dirichletovi uvjeti, koje ovdje ne treba testirati), Fourierov red konvergira ka vrijednosti $f(x)$ u svim tačkama x unutar intervala (p, q) kojima je funkcija f neprekidna (čisto da se zna, ukoliko je x tačka prekida, Fourierov red tada konvergira ka vrijednosti $(f(x+0) - f(x-0))/2$, a u rubnim tačkama intervala, Fourierov red konvergira ka vrijednosti $(f(p) + f(q))/2$). To ujedno znači da pod istim uvjetima, Fourierov polinom dobro aproksimira funkciju f unutar intervala (p, q), tim bolje što je stepen N veći (van intervala (p, q) Fourierov red konvergira ka funkciji koja se dobija kada se funkcija f periodički produži izvan tog intervala). Naravno, prethodne formule za a_k i b_k nepodesne su za računanje na računaru, jer se u njima javlja integral. Srećom, postoje formule kojima se integrali mogu sasvim lijepo približno izračunati koristeći samo elementarne računske operacije. Primjena jedne od takvih formula na gornje integrale (poznate kao trapezno pravilo) daje sljedeće aproksimativne formule:

$$a_k \approx \frac{f(p) + f(q)}{\mathrm{M}} \cos\frac{2k\pi p}{\mathrm{T}} + \frac{2}{\mathrm{M}} \sum_{\mathrm{i}=1}^{\mathrm{M}-1} f\left(p + \frac{i\mathrm{T}}{\mathrm{M}}\right) \cos\left[2k\pi\left(\frac{p}{\mathrm{T}} + \frac{i}{\mathrm{M}}\right)\right]$$

$$b_k \approx \frac{f(p) + f(q)}{\mathrm{M}} \sin\frac{2k\pi p}{\mathrm{T}} + \frac{2}{\mathrm{M}} \sum_{\mathrm{i}=1}^{\mathrm{M}-1} f\left(p + \frac{i\mathrm{T}}{\mathrm{M}}\right) \sin\left[2k\pi\left(\frac{p}{\mathrm{T}} + \frac{i}{\mathrm{M}}\right)\right]$$

Upravo ove formule ćete koristiti za računanje koeficijenata a_k i b_k iz zadane funkcije f. Ovdje je M parametar (tzv. broj podintervala) od kojeg zavisi tačnost računanja. Što je M veći, tačnost je bolja (za postizanje dobre tačnosti, M bi trebao biti barem nekoliko stotina, ili više). U slučaju da N nije pozitivan, baca se izuzetak tipa "`domain_error`" uz prateći tekst "Stepen mora biti pozitivan". Isti tip izuzetka, samo uz prateći tekst "Broj podintervala mora biti pozitivan" baca se ukoliko M nije pozitivan, dok se ukoliko nije ispunjeno $p < q$ baca izuzetak tipa "`range_error`" uz prateći tekst "Neispravan interval".

- Preklopljeni operator "`()`" koji omogućava računanje Fourierovog polinoma u zadanoj tački. Na primjer, ukoliko je "`f`" neki objekat tipa "`FourierovRed`", tada "`f(2)`" treba da predstavlja vrijednost odgovarajućeg Fourierovog polinoma u tački $x = 2$.
- Preklopljeni operator "`[]`" koji omogućava pristup koeficijentima Fourierovog polinoma. Ukoliko je "`f`" neki objekat tipa "`FourierovRed`", a "`k`" neki nenegativni cijeli broj k, tada "`f[k]`" treba da dâ uređeni par (tj. objekat tipa "`std::pair<double, double>`" koji se sastoji od koeficijenata a_k i b_k. U slučaju da je k nula, treba smatrati da je $b_0 = 0$, a u slučaju da je k negativan ili veći od stepena N, treba baciti izuzetak tipa tipa "`range_error`" uz prateći tekst "Neispravan indeks". Pored toga, operator "`[]`" treba da, ukoliko se primijeni na nekonstantne objekte tipa "`FourierovRed`", omogući ne samo čitanje nego i promjenu koeficijenata a_k i b_k. Recimo, ukoliko je "`f`" neki nekonstantni Fourierov polinom čiji je stepen barem 3, tada konstrukcija poput

```
f[3] = std::make_pair(0.7, 3.41);
```

treba da postavi koeficijente a_3 i b_3 na 0.7 i 3.41 respektivno. U ovom slučaju, ukoliko se zada $k = 0$, drugu koordinatu uređenog para treba ignorirati (jer koeficijenta b_0 nema).

Za smještanje koeficijenata a_k i b_k trebate koristiti dinamički alocirane nizove realnih brojeva. Samim tim, potrebno je da u klasi predvidite sve upravljačke komponente koji će obezbijediti automatsku alokaciju memorije, te sigurno kopiranje i međusobno dodjeljivanje primjeraka ove klase, pri čemu izvedbe trebaju biti optimizirane u slučaju da se kopiraju privremeni objekti.

Napisanu klasu demonstrirajte u testnom programu u kojem ćete testirati sve elemente klase (ne zaboravite testirati i da upravljačke komponente rade ispravno). Posebno ćete obaviti testiranje na primjeru razvoja funkcije $f(x) = x^2$ u Fourierov red na intervalu $(0, 2\pi)$. Tačni koeficijenti razvoja ove funkcije u Fourierov red na zadanom intervalu glase $a_0 = 8\pi^2/3$, $a_k = 4/k^2$ za $k \neq 0$ i $b_k = -4\pi/k$. Kreirajte jedan objekat preko tačnih vrijednosti koeficijenata a drugi u kojem se ovi koeficijenti računaju aproksimativno, pa uporedite kakav je stepen slaganja za različite vrijednosti broja podintervala M. Također uporedite tačnu vrijednost funkcije f u nekoj tački unutar intervala $(0, 2\pi)$ (recimo $x = 3$) sa vrijednošću koja se dobije iz Fourierovog polinoma za razne vrijednosti stepena N (za veće vrijednosti N, recimo za $\mathrm{N} > 50$, razlika ne bi trebala biti veća od nekoliko procenata, osim u tačkama blizu krajeva intervala, gdje greška raste).

Napomena: potencijalni inženjer za kojeg je $\pi = 3.14$ ne zaslužuje da ikada postane inženjer.

4. Prepravite klasu iz prethodnog zadatka tako da se za smještanje koeficijenata umjesto dva dinamički alocirana niza koriste dva vektora (izvršite sve neophodne izmjene koje su potrebne da bi klasa radila identično kao klasa iz prethodnog zadatka).

5. Za lakše praćenje stanja dionica na nekoj berzi, potrebno je definirati i implementirati klasu nazvanu "`Berza`". Ova klasa omogućava čuvanje dnevnih informacija o tekućoj cijeni dionica za izvjesni vremenski period u vektoru cijelih brojeva, kojem se pristupa preko odgovarajućeg privatnog atributa (cijene se zadaju u feninzima, tako da se mogu iskazati kao cijeli broj). Interfejs klase sadrži sljedeće elemente:

   - Konstruktor sa dva parametra koji predstavljaju minimalnu i maksimalnu cijenu koja se može registrirati, te konstruktor sa jednim parametrom koji je maksimalna cijena koja se može registrirati (minimalna cijena se tada postavlja na nulu). Ovi parametri moraju biti pozitivni, u suprotnom treba baciti izuzetak tipa "`range_error`" uz prateći tekst "Ilegalne granicne cijene". Konstruktor sa jednim parametrom ne smije se koristiti za automatsku konverziju cjelobrojnih podataka u objekte tipa "`Berza`".
   - Metodu "`RegistrirajCijenu`" koja vrši registraciju nove cijene dionice zadane putem parametra. U slučaju da je cijena manja ili veća od minimalne odnosno maksimalne dozvoljene cijene, treba baciti izuzetak tipa "`range_error`" uz prateći tekst "Ilegalna cijena".
   - Metodu "`DajBrojRegistriranihCijena`" koja daje broj registriranih cijena.
   - Metodu "`BrisiSve`" koja briše sve unesene cijene.
   - Metode "`DajMinimalnuCijenu`" i "`DajMaksimalnuCijenu`" koje vraćaju kao rezultat minimalnu i maksimalnu registriranu cijenu (u slučaju da nema registriranih cijena, obje metode treba da bace izuzetak tipa "`range_error`" uz prateći tekst "Nema registriranih cijena"). Za realizaciju nije dozvoljeno koristiti petlje, nego isključivo odgovarajuće funkcije iz biblioteke "`algorithm`".
   - Unarni operator "`!`" koji primijenjen na objekat daje logičku vrijednost "tačno" ukoliko nema niti jedne registrirane cijene, u suprotnom daje logičku vrijednost "netačno".
   - Metodu "`DajBrojCijenaVecihOd`" koja vraća kao rezultat broj cijena većih od vrijednosti koja se zadaje kao parametar (u slučaju da nema registriranih cijena, metode treba da bace izuzetak tipa "`range_error`" uz prateći tekst "Nema registriranih cijena"). 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).
   - Metodu "`Ispisi`" koja ispisuje sve unesene (registrirane) cijene sortirane u *opadajućem poretku* (tj. najveća cijena se ispisuje prva), pri čemu se svaka cijena ispisuje u posebnom redu. Cijene treba ispisivati u KM (ne u feninzima), pri čemu se uvijek prikazuju dvije decimale. Pri tome je neophodno koristiti funkcije i/ili funktore iz biblioteka "`algorithm`" i "`functional`" (upotreba lambda funkcija ili pomoćnih imenovanih funkcija nije dozvoljena). Ova funkcija obavezno treba biti inspektor funkcija, tj. treba biti deklarirana sa modifikatorom "**`const`**"!
   - Preklopljeni operator "`[]`" koji omogućava da se direktno pročita *i*-ta registrirana cijena (numeracija ide od jedinice). Ukoliko je indeks izvan dozvoljenog opsega, treba baciti izuzetak tipa "`range_error`" uz prateći tekst "Neispravan indeks". Pri tome, ovaj operator se *ne može koristiti* za izmjenu podataka, odnosno ne može se koristiti sa lijeve strane znaka jednakosti.
   - Preklopljene operatore "`++`" i "`--`" koji povećavaju odnosno smanjuju sve registrirane cijene za 1 KM. U slučaju da pri tome neka od cijena premaši maksimalno dozvoljenu vrijednost ili podbaci minimalno dozvoljenu vrijednost, baca se izuzetak tipa "`range_error`" uz prateći tekst "Prekoracen dozvoljeni opseg cijena". Potrebno je podržati kako prefiksnu, tako i postfiksnu verziju ovih 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 i ranije).
   - Preklopljeni unarni operator "`-`" koji daje kao rezultat novi objekat tipa "`Berza`" u kojem su sve cijene oduzete od zbira maksimalno i minimalno dozvoljene cijene (ovim sve cijene i dalje sigurno ostaju u dozvoljenom opsegu). 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 i ranije).

- Preklopljene operatore "`+`" i "`-`" koji djeluju na sljedeći način. Ukoliko je "`X`" objekat tipa "`Berza`", a "`Y`" cijeli broj, tada je "`X + Y`" je novi objekat tipa "`Berza`" u kojem su sve registrirane cijene povećane za iznos "`Y`". Slično, "`X - Y`" je novi objekat tipa "`Berza`" u kojem su sve registrirane cijene 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 cijene oduzete od vrijednosti "`Y`". Ukoliko se kao rezultat ovih operacija dogodi da neka od cijena izađe izvan dozvoljenog opsega, treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Prekoracen dozvoljeni opseg cijena". U slučaju kada su i "`X`" i "`Y`" objekti tipa "`Berza`", tada je izraz "`X - Y`" novi objekat tipa "`Berza`" koji sadrži *razlike* odgovarajućih cijena iz objekata "`X`" i "`Y`". U ovom posljednjem slučaju se podrazumijeva da "`X`" i "`Y`" sadrže isti broj registriranih cijena, te da imaju iste minimalne i maksimalne dozvoljene cijene (u suprotnom, treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Nesaglasni operandi"). Ukoliko pri tome bilo koja od razlika izađe izvan dozvoljenog opsega cijena, treba baciti isti izuzetak kao i u prethodnim slučajevima. U svim ostalim situacijama, izrazi "`X + Y`" odnosno "`X - Y`" ne trebaju biti sintaksno ispravni. Za realizaciju ovih operatora nije dozvoljeno koristiti petlje, niti lambda funkcije ili pomoćne imenovane funkcije, nego isključivo prikladne funkcije i/ili funktore iz biblioteka "`algorithm`" i "`functional`" (vrijedi ista napomena kao i ranije).
- 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.
- Preklopljene relacione operatore "`==`" i "`!=`" koje ispituju da li su dva objekta tipa "`Berza`" jednaka ili nisu. Dva objekta ovog tipa smatraju se jednakim ukoliko sadrže isti broj registriranih cijena, i ukoliko su sve odgovarajuće registrirane cijene u oba objekta jednake. Za realizaciju ovih operatora nije dozvoljeno koristiti petlje, nego isključivo odgovarajuće funkcije i/ili funktore iz biblioteka "`algorithm`" i "`functional`".

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.