Zadaci za Tutorijal 12.

NAPOMENA: Studenti bi trebali da razmisle o zadacima koji će se raditi na tutorijalu prije nego što dođu na tutorijal, tako da već u startu imaju osnovne ideje kako riješiti zadatke. U suprotnom, rad na laboratorijskim vježbama neće biti produktivan. Činjenica je da će mali broj studenata stići uraditi sve zadatke za vrijeme koje je predviđeno za tutorijal. Međutim, svim studentima se toplo savjetuje da dovrše samostalno one zadatke koje nisu stigli uraditi za vrijeme tutorijala, s obzirom da je razumijevanje ovog tipa zadataka od vitalne važnosti za polaganje drugog parcijalnog ispita.

1. Izmijenite klasu "`Sat`" razvijenu u Zadatku 4. sa Tutorijala 9. tako da se umjesto metoda "`Sljedeci`" i "`Prethodni`" koriste operatori "`++`" i "`--`" (pri čemu je potrebno podržati kako prefiksne, tako i postfiksne verzije ovih operatora), umjesto metode "`PomjeriZa`" operatori "`+=`" i "`-=`" (pri čemu "`s -= n`" ima isto značenje kao i "`s += -n`", umjesto metode "`Ispisi`" operator "`<<`", a umjesto funkcije "`BrojSekundiIzmedju`" ili funkcije "`Razmak`" operator "`-`". Također podržite da su mogući i izrazi oblika "`s + n`" i "`s - n`" gdje je "`s`" objekat tipa "`Sat`" a "`n`" cijeli broj koji imaju slično dejstvo kao izrazi "`s += n`" i "`s -= n`", samo što oni ne modificiraju objekat "`s`" nego vraćaju kao rezultat novi objekav tipa "`Sat`" nastao pomjeranjem vremena u objektu "`s`" za "`n`" sekundi unaprijed odnosno unazad. Testni program prilagodite novom interfejsu klase. Radi jednostavnijeg rada, dodajte ovoj klasi konstruktor bez parametara, koji kreira sat inicijaliziran na 00:00:00, kao i konstruktor sa 3 parametra koji obavlja istu stvar kao i metoda "`PostaviNormalizirano`", samo odmah prilikom kreiranja objekta.

2. Za skupovni tip "`set`" iz istoimene biblioteke definirajte operatore "`+`" i "`*`" koji će za dva skupa sa istim tipom elemenata dati kao rezultat respektivno njihovu uniju i presjek, kao i operator "`<<`" koji omogućava ispis elemenata skupa u vitičastim zagradama, koji su međusobno razdvojeni zarezima (možete se poslužiti Zadatkom 7 iz Tutorijala 7). Također definirajte i odgovarajuće operatore "`+`" i "`+=`" i "`*=`" Nakon definiranja ovih operatora, mora postati moguće nešto poput sljedećeg:

```
std::set<int> s1{3, 5, 1, 2, 8}, s2{7, 2, 8, 4};
std::cout << s1 + s2 << std::endl;                  // Ispis {1,2,3,4,5,7,8}
std::cout << s1 * s2 << std::endl;                  // Ispis {2,8}
```

   Također definirajte i operator "`%`" koji daje kao rezultat Dekartov proizvod skupova-operanada, pri čemu tip njihovih elemenata ne mora biti isti (operator "`%=`" ovdje nije moguće definirati, razmislite zašto). Kako su elementi Dekartovog proizvoda zapravo uređeni parovi, za njihovo formiranje koristite bibliotečki tip "`pair`". Dalje, kako za objekte tipa "`pair`" nije inicijalno uvedena podrška za ispisivanje na ekran, uvedite i tu podršku (preklapanjem operatora "`<<`") da omogućite njihov ispis na ekran kao uređenih parova u zagradama, sa prvom i drugom koordinatom razdvojenom zarezom. Kad sve ovo uradite, trebalo bi postati moguće nešto poput sljedećeg:

```
std::set<char> s3{'A', 'B'};
std::set<char> s4{1, 2, 3};
std::cout << s1 % s2 << std::endl;    // Ispis {(A,1),(A,2),(A,3),(B,1),(B,2),(B,3)}
```

3. Neka je dat pobrojani tip

```
enum Dani {Ponedjeljak, Utorak, Srijeda, Cetvrtak, Petak, Subota, Nedjelja};
```

   Poznato je da se, u odsustvu drugačijih uputa, promjenljive tipa "`Dani`" automatski konvertiraju u cjelobrojne vrijednosti. Recimo, ukoliko promjenljiva "`d`" tipa "`Dani`" ima vrijednost "`Srijeda`", naredba poput "`std::cout << d`" će umjesto teksta "Srijeda" ispisati broj 2. Srećom, ovakvo ponašanje se može izmijeniti korištenjem preklapanja operatora. Napišite operatorsku funkciju za operator "`<<`" koji će omogućiti da se prilikom ispisa konstanti, promjenljivih i izraza tipa "`Dani`" umjesto brojčane vrijednosti dobijene konverzijom ispisuje tekst koji odgovara njihovom značenju. Testirajte napisanu funkciju u testnom programu koji sadrži petlju

```
for(Dani d = Ponedjeljak; d <= Petak; d++) std::cout << d << std::endl;
```

   Takva petlja treba da zaista ispiše imena dana "Ponedjeljak", "Utorak", "Srijeda", "Četvrtak" i "Petak", a ne brojeve od 0 do 4. Za tu svrhu ćete također morati definirati i operator "`++`" koji podrazumijevano nije definiran za promjenljive pobrojanog tipa (uradite to na način kako je urađeno u Predavanju 12_a).

Kada to uradite, probajte napraviti sličnu petlju koja bi trebala da ispiše sve dane u sedmici redom od ponedjeljka do nedjelje. U čemu je problem? Zbog čega petlja ne radi kako je očekivano? Šta se može učiniti po tom pitanju? Riješite ovaj problem, odnosno napravite petlju koja će korektno ispisati sve dane u sedmici.

4. Prepravite operator "`<<`" u klasi "`Kompleksni`" sa Predavanja 12_a tako da se ispis kompleksnih brojeva vrši u algebarskom formatu, tj. u obliku poput "`2+3i`" umjesto "`(2,3)`". Obratite pažnju na razne specijalne slučajeve koji mogu nastupiti, tako da recimo treba ispisivati "`2-3i`", "`3+i`", "`1-i`", "`2i`", "`-5i`", "`i`", "`3`" itd. a ne "`2+-3i`", "`3+1i`", "`1-1i`" (ili, još gore, "`1+-1i`"), "`0+2i`", "`0-5i`" (ili, još gore, "`0+-5i`"), "`0+1i`" (ili "`0+i`"), "`3+0i`", itd. Kao dodatni izazov (ovo se neće testirati na tutorijalu), probajte prepraviti operator "`>>`" da omogući unos kompleksnih brojeva sa tastature u istom formatu kao prilikom ispisa. Treba da se prihvataju sve smislene varijante unosa koje se mogu protumačiti kao kompleksan broj. Uspijete li ovo uraditi, *obavezno se javite predmetnom nastavniku*.

5. Napravite program koji traži da se sa tastature unese slijed realnih brojeva u neki vektor (broj elemenata se također unosi sa tastature), a zatim zamjenjuje sve elemente vektora njihovim recipročnim vrijednostima i ispisuje elemente tako transformiranog vektora brojeva na ekran. Pri tome, transformaciju elemenata niza treba izvesti *jednom jedinom naredbom*, bez upotrebi petlji. Također, u programu nije dozvoljeno definirati *nikakve pomoćne funkcije* (dakle, "`main`" treba da bude *jedina funkcija u programu*), pa čak ni anonimne lambda funkcije. Za tu svrhu, poslužite se odgovarajućim funkcijama i funktorima iz biblioteka "`algorithm`" i "`functional`". Dijalog između korisnika i programa treba da izgleda poput sljedećeg:

```
Koliko zelite elemenata: 5
Unesite elemente: 1 2 3 4 5
Transformirani elementi: 1 0.5 0.333333 0.25 0.2
```

Uputa: prvo probajte za rješavanje problema iskoristiti funkciju "`bind1st`". Kada riješite problem pomoću ove funkcije, zamijenite je sa funkcijom "`bind`", s obzirom da je ona proglašena zastarjelom i njena upotreba se više ne preporučuje (štaviše, u standardu C++17 biće skroz izbačena).

6. Prepravite generičku funkciju "`Matrica`" sa Predavanja 11_b tako da se umjesto funkcija članica "`Unesi`" i "`Ispisi`" te funkcije "`ZbirMatrica`" koriste odgovarajući operatori. Konkretno, dio "`main`" funkcije unutar *try*-bloka nakon prepravke trebao bi izgledati ovako:

```
Matrica<double> a(m, n, 'A'), b(m, n, 'B');
std::cout << "Unesi matricu A:\n";
std::cin >> a;
std::cout << "Unesi matricu B:\n";
std::cin >> b;
std::cout << "Zbir ove dvije matrice je:\n";
std::cout << std::setw(7) << a + b;
```

Ovdje je jedino "misteriozno" kako postići da operatorska funkcija za operator "`<<`" sazna koja je širina postavljena manipulatorom "`setw`". Za tu svrhu, unutar operatorske funkcije nad objektom toka treba pozvati funkciju "`width`" bez parametara, i ona će kao rezultat dati trenutno postavljenu širinu ispisa (tj. onu koja je prethodno postavljena). Nakon obavljenih prepravki dodajte podršku za još neke operatore. Na prvom mjestu, dodajte podršku za operator "`-`" i "`*`". Operator "`*`" treba da podrži kako množenje dvije matrice, tako i množenje matrice a brojem i obrnuto. Ukoliko matrice nisu saglasne za množenje, treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Matrice nisu saglasne za množenje". Zatim, kako programerski bonton nalaže, dodajte podršku za operatore "`+=`" i "`-=`" i "`*=`", koji će obezbijediti da izrazi oblika "`A += B`", "`A -= B`" i "`A *= B`" uvijek imaju logički isto značenje kao i izrazi "`A = A + B`", "`A = A - B`" i "`A = A * B`" kad god to ima smisla. Dalje, omogućite da se elementima matrice može pristupiti konstrukcijama poput "`A[i][j]`" i "`A(i,j)`", pri čemu prva konstrukcija vrši pristup u C/C++ stilu, sa indeksacijom od nule i bez provjere ispravnosti indeksa), dok druga konstrukcija vrši pristup u matematičkom stilu, sa indeksacijom od jedinice i uz provjeru ispravnosti indeksa (u slučaju neispravnosti, baca se izuzetak tipa "`range_error`" uz prateći tekst "Neispravan indeks". Konstrukcije poput "`A[i][j]`" i "`A(i,j)`" moraju se moći koristiti i sa lijeve strane znaka jednakosti, ali samo nad nekonstantnim objektima tipa "`Matrica`". Konačno, treba podržati i automatsku konverziju objekata tipa "`Matrica`" u tip "`string`", tako da se kao rezultat konverzije dobija string koji sadrži elemente matrice unutar vitičastih zagrada međusobno razdvojene zarezima (npr. "`{{1,2,3},{4,5,6}}`"). Obavezno napišite i testni program u kojem ćete demonstrirati implementirane operatore.