Zadaća 2

Ova zadaća nosi **5 poena** i predstavlja *nadogradnju zadaće 1*. Rok za predaju je **ponedjeljak**, **27. XI 2023.** do kraja dana. Zadaće se predaju putem Zamgera. Broj poena koji student dobija ovisi od broja korektno implementiranih stavki koje se traže u zadaći.

Vaš zadatak je da dopunite klase "`Vector`" i "`Matrix`" razvijene u prethodnoj zadaći s nekoliko novih funkcionalnosti), kao i da razvijete dvije nove klase "`LUDecomposer`" i "`QRDecomposer`".

Nove stvari koje treba dodati u interfejs klase "`Vector`" su sljedeće:

```
void Chop(double eps = -1);
bool EqualTo(const Vector &v, double eps = -1) const;
```

Funkcija članica "`Chop`" sve elemente vektora na koji je primijenjena, a koji su manji od iznosa "praga odsjecanja" zadanog putem parametra "`eps`", postavlja na nulu. Ukoliko ovaj parametar ima negativnu vrijednost, kao prag odsjecanja se uzima vrijednost dobijena pozivom funkcije "`GetEpsilon`". Kako ovaj parametar ima podrazumijevanu vrijednost -1, upravo će se takav prag podrazumijevati ukoliko se parametar izostavi. Glavna primjena ove funkcije je da se nakon nekog složenijeg računa elementi vektora koji bi zapravo trebali biti jednaki nuli, a nisu ispali jednaki nuli zbog akumuliranih grešaka u računu, zaista svedu na nulu.

Funkcija članica "`EqualTo`" testira da li je vektor na koji je primijenjena jednak vektoru koji joj je prenesen kao prvi parametar, i vraća kao rezultat logičku vrijednost "tačno" ili "netačno", ovisno od rezultata poređenja. Međutim, umjesto testiranja na striktnu jednakost, ova funkcija dopušta i da se elementi vektora koji se porede neznatno razlikuju. Tačnije, ova funkcija smatra da su dva vektora jednaka ukoliko su iste dimenzije i ukoliko im se odgovarajući elementi razlikuju za manje od tolerancije zadane putem parametra "`eps`". Ukoliko ovaj parametar ima negativnu vrijednost, kao tolerancija se uzima vrijednost dobijena pozivom funkcije "`GetEpsilon`". Kako ovaj parametar ima podrazumijevanu vrijednost -1, upravo će se ta tolerancija podrazumijevati ukoliko se parametar izostavi.

Nove stvari koje treba dodati u interfejs klase "`Matrix`" su sljedeće:

```
void Chop(double eps = -1);
bool EqualTo(const Matrix &m, double eps = -1) const;
friend Matrix LeftDiv(Matrix m1, Matrix m2);
friend Vector LeftDiv(Matrix m, Vector v);
friend Matrix operator /(const Matrix &m, double s);
Matrix &operator /=(double s);
friend Matrix operator /(Matrix m1, Matrix m2);
Matrix &operator /=(Matrix m);
double Det() const;
friend double Det(Matrix m);
void Invert();
friend Matrix Inverse(Matrix m);
void ReduceToRREF();
friend Matrix RREF(Matrix m);
int Rank() const;
friend int Rank(Matrix m);
```

Funkcije "`Chop`" i "`EqualTo`" za klasu "`Matrix`" potpuno su analogne istoimenim funkcijama u klasi "`Vector`", samo što se odnose na elemente matrice a ne vektora, tako da one ne traže nikakva posebna objašnjenja. Naravno, jasno je da se jednakim mogu smatrati samo matrice čije se obje dimenzije podudaraju.

Funkcija "`LeftDiv`" računa matrični lijevi količnik $\boldsymbol{M}_1 \backslash \boldsymbol{M}_2$ dvije matrice $\boldsymbol{M}_1$ i $\boldsymbol{M}_2$, tj. nalazi rješenje za $\boldsymbol{X}$ matrične jednačine $\boldsymbol{M}_1\boldsymbol{X} = \boldsymbol{M}_2$ i vraća matricu $\boldsymbol{X}$ kao rezultat. Podržano je da drugi operand bude vektor, što je pogodno za klasično rješavanje sistema linearnih jednačina (u tom slučaju, rezultat je također vektor). Račun treba obaviti klasičnom Gaussovom eliminacijom sa djelimičnom pivotizacijom. Parametri ove funkcije se ne prenose kao reference, s obzirom da klasična Gaussova eliminacija uništava sadržaj matrica, pa bismo svakako trebali praviti kopiju argumenata, a ovako se kopija kreira automatski. U slučaju da matrica $\boldsymbol{M}_1$ nije kvadratna, treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Divisor matrix is not square". U slučaju da je matrica $\boldsymbol{M}_1$ singularna, treba baciti izuzetak tipa

"domain_error" uz prateći tekst "Divisor matrix is singular". U slučaju da format matrice $\boldsymbol{M}_2$ nije odgovarajući (različit broj redova u odnosu na matricu $\boldsymbol{M}_1$), treba baciti izuzetak tipa "domain_error" uz prateći tekst "Incompatible formats".

Operator "/" treba da podrži dijeljenje matrice skalarom, ali također treba biti podržano i desno matrično dijeljenje, odnosno $\boldsymbol{M}_1/\boldsymbol{M}_2$ za slučaj kada su $\boldsymbol{M}_1$ i $\boldsymbol{M}_2$ matrice treba da bude rješenje po $\boldsymbol{X}$ matrične jednačine $\boldsymbol{X}\,\boldsymbol{M}_2 = \boldsymbol{M}_1$. Pored operatora "/", treba podržati i verziju operatora sa dodjeljivanjem "/=". Pri tome je, zbog same prirode postupka kako se ove operacije obavljaju, mudra ideja prvo realizirati operator "/=", a onda operator "/". Pri tome, bez obzira na vezu $\boldsymbol{M}_1/\boldsymbol{M}_2 = (\boldsymbol{M}_2^{\mathrm{T}} \backslash \boldsymbol{M}_1^{\mathrm{T}})^{\mathrm{T}}$, *nemojte realizirati ove operatore pozivom funkcija* "Transpose" *i* "LeftDiv", *nego obavite odgovarajuće izmjene u algoritmu za matrično lijevo dijeljenje* (*transpozicije indeksa*). Vodite samo računa da ćete ovdje prilikom pivotizacije razmjenjivati *kolone*, a ne redove. Vrijede slične napomene vezane za izuzetke kao i za funkciju "LeftDiv" (nemojte brzopleto zaključivati kada treba prijaviti grešku "Incompatible formats"). Za slučaj dijeljenja skalarom, u slučaju da je djelilac 0, treba baciti izuzetak tipa "domain_error" uz prateći tekst "Division by zero".

Funkcija "Det" vraća kao rezultat determinantu matrice. Podržane su dvije verzije: funkcija članica, te klasična funkcija, kojoj se matrica čija se determinanta računa prenosi kao parametar. U slučaju da matrica nije kvadratna, treba baciti izuzetak tipa "domain_error" uz prateći tekst "Matrix is not square".

Funkcija članica "Invert" treba da invertira matricu na koju je primijenjena, odnosno nakon poziva ove funkcije članice, matrica na koju je ova funkcija članica primijenjena treba da se izmijeni tako da postane jednaka inverznoj matrici matrice koja je bila prije poziva. Računanje treba obaviti pomoću Gauss-Jordanove eliminacije koja obavlja inverziju "skoro u mjestu", odnosno bez alociranja ikakvih dodatnih matrica, vektora itd. osim jednog vektora cijelih brojeva, koji će služiti za vođenje evidencije o izvršenim razmjenama redova tokom pivotizacije. Za razliku od ove funkcije članice, klasična funkcija "Inverse" daje kao rezultat inverznu matricu matrice koja joj se prenosi kao parametar. Za obje funkcije, u slučaju da matrica nije kvadratna, treba baciti izuzetak tipa "domain_error" uz prateći tekst "Matrix is not square". U slučaju da je matrica singularna, treba baciti izuzetak tipa "domain_error" uz prateći tekst "Matrix is singular".

Funkcija članica "ReduceToRREF" transformira matricu na reducirani oblik ustrojen po redovima koristeći modifikaciju Gauss-Jordanove eliminacije. Slična je klasična funkcija "RREF", koja daje kao rezultat reducirani oblik ustrojen po redovima matrice koja joj je prenesena kao parametar. U vezi sa ovim funkcijama su i funkcija članica "Rank", koja vraća kao rezultat rang matrice (koji se određuje uz pomoć transformacije na reducirani oblik ustrojeni po redovima), kao i istoimena klasična funkcija, koja vraća kao rezultat rang matrice koja joj je prenesena kao parametar.

U funkcijama koje treba napisati ima dosta zajedničkog kôda. Da bi se izbjeglo kopiranje kôda, dozvoljeno je kreirati razne pomoćne funkcije, koje onda treba smjestiti u privatni dio klase.

Interfejs klase "LUDecomposer", sastoji se sljedećih elemenata:

```
LUDecomposer(Matrix m);
void Solve(const Vector &b, Vector &x) const;
Vector Solve(Vector b) const;
void Solve(const Matrix &b, Matrix &x) const;
Matrix Solve(Matrix b) const;
Matrix GetCompactLU() const;
Matrix GetL() const;
Matrix GetU() const;
Vector GetPermuation() const;
```

Konstruktor klase treba da izvrši LU faktorizaciju matrice (Doolittleovog tipa) koja joj je prenesena kao parametar korištenjem Croutovog algoritma sa pivotizacijom. Tom prilikom, rezultati faktorizacije i informacije o obavljenim razmjenama redova pohranjuju se negdje u privatnim atributima klase. U slučaju da matrica nije kvadratna, treba baciti izuzetak tipa "domain_error" uz prateći tekst "Matrix is not square". U slučaju da je matrica singularna, treba baciti izuzetak tipa "domain_error" uz prateći tekst "Matrix is singular".

Nakon što je obavljena LU faktorizacija, pozivom metode "Solve" mogu se rješavati sistemi linearnih jednačina sa različitim desnim stranama u vremenu reda $O(n^2)$ umjesto $O(n^3)$ gdje je n broj jednačina. Prva varijanta metode "Solve" kao prvi parametar prima vektor slobodnih koeficijenata sa desne strane

jednačina, a kao drugi parametar vektor u koji se smješta rješenje. Dozvoljeno je da oba parametra budu isti vektor, čime se prosto rješenje prepisuje preko vektora slobodnih koeficijenata. Druga varijanta prima kao parametar vektor slobodnih koeficijenata, a vraća kao rezultat vektor rješenja. Treća i četvrta varijanta analogne su prvoj i drugoj varijanti, ali omogućavaju da se umjesto vektora slobodnih članova zada matrica, tako da je moguće rješavati i matrične jednačine oblika $\boldsymbol{A}\,\boldsymbol{X} = \boldsymbol{B}$. U svim slučajevima, u slučaju da dimenzije parametara nisu odgovarajuće, treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Incompatible formats".

Podržane su i pristupne metode pomoću kojih je moguće pristupiti informacijama o obavljenoj faktorizaciji. Metoda "`GetCompactLU`" daje kao rezultat matricu koja u sebi sadrži sve netrivijalne elemente obje matrice $\boldsymbol{L}$ i $\boldsymbol{U}$ (elementi matrice $\boldsymbol{L}$ su ispod dijagonale, a elementi matrice $\boldsymbol{U}$ na dijagonali i iznad nje). Ovo je zapravo oblik koji se dobija kao rezultat standardnih izvedbi algoritama za LU faktorizaciju. Metode "`GetL`" i "`GetU`" daju kao rezultat kompletne matrice $\boldsymbol{L}$ odnosno $\boldsymbol{U}$, što uključuje i njihove trivijalne elemente, za koje se zna da su nule (odnosno jedinice na dijagonali matrice $\boldsymbol{L}$). Konačno, metoda "`GetPermutation`" vraća kao rezultat permutacioni vektor koji sadrži informacije o obavljenim razmjenama redova koje su izvršene usljed pivotizacije (mada su svi elementi ovog vektora prirodni brojevi, rezutat treba ipak smjestiti u objekat tipa "`Vector`" čiji su elementi realni). Pri tome, te informacije trebaju biti u skladu s matematičkom konvencijom, po kojoj su redovi i kolone numerirani od jedinice.

Interfejs klase "`QRDecomposer`", sastoji se sljedećih elemenata:

```
QRDecomposer(Matrix m);
void Solve(const Vector &b, Vector &x) const;
Vector Solve(Vector b) const;
void Solve(Matrix &b, Matrix &x) const;
Matrix Solve(Matrix b) const;
Vector MulQWith(Vector v) const;
Matrix MulQWith(Matrix m) const;
Vector MulQTWith(Vector v) const;
Matrix MulQTWith(Matrix m) const;
Matrix GetQ() const;
Matrix GetR() const;
```

Konstruktor klase treba da izvrši nepotpunu QR faktorizaciju matrice koja joj je prenesena kao parametar korištenjem Householderovog algoritma. Faktorizacija je nepotpuna u smislu da se matrica $\boldsymbol{Q}$ ne formira eksplicitno, nego se umjesto toga čuvaju odgovarajući vektori korišteni za formiranje Householderovih matrica (iz kojih se mogu izvući sve neophodne informacije koje bi se mogle dobiti i iz matrice $\boldsymbol{Q}$). Sve informacije o obavljenoj faktorizaciji pohranjuju se negdje u privatnim atributima klase. Pri tome, utrošak memorije treba biti što je god moguće manji, odnosno sve što je neophodno treba čuvati samo u jednoj matrici koja izmješano sadrži elemente matrice $\boldsymbol{R}$ iznad gornje dijagonale i elemente pomoćnih vektora, te jednom dodatnom vektoru koji sadrži dijagonalne elemente matrice $\boldsymbol{R}$. Matrica ne mora biti kvadratna, ali broj njenih redova mora biti veći ili jednak od broja kolona, u suprotnom treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Invalid matrix format". U slučaju da je matrica singularna, treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Matrix is singular" (singularnost se može prepoznati ukoliko se u toku algoritma pojavi dijeljenje nulom, odnosno brojem čija je apsolutna vrijednost manja od dozvoljene tolerancije ε).

Nakon što je obavljena QR faktorizacija, i ovdje se pozivom metode "`Solve`" mogu se rješavati sistemi linearnih jednačina sa različitim desnim stranama u vremenu reda $O(n^2)$ umjesto $O(n^3)$ gdje je n broj jednačina. Za ove metode vrijedi potpuno isti opis kao u slučaju klase "`LUDecomposer`". Pri tome, ove metode trebaju raditi samo u slučaju da je faktorizirana matrica bila kvadratna (tj. ne treba podržavati aproksimativno rješavanje preodređenih sistema). U slučaju da to nije bilo tako, treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Matrix is not square".

Metode "`MulQWith`" i "`MulQTWith`" služe za manipulacije uz pomoć matrice $\boldsymbol{Q}$, koja nije eksplicitno formirana. Metoda "`MulQWith`" u dvije varijante računa produkt matrice $\boldsymbol{Q}$ sa vektorom odnosno matricom koji su zadani kao parametri, bez eksplicitnog formiranja matrice $\boldsymbol{Q}$, nego samo koristeći informacije pohranjene tokom faktorizacije. Metoda "`MulQTWith`" radi istu stvar, samo računa produkt transponirane matrice $\boldsymbol{Q}^{\mathrm{T}}$ sa vektorom odnosno matricom koji su zadani kao parametri. U svim slučajevima, ukoliko parametar nema ispravan format da bi se moglo izvršiti množenje, treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Incompatible formats".

Konačno, treba podržati i metodu "`GetQ`" koja ekplicitno formira matricu $\boldsymbol{Q}$ i vraća je kao rezultat, kao i metodu "`GetR`" koja vraća kompletnu matricu $\boldsymbol{R}$ onakvu kakva ona treba zaista da bude (sa nulama ispod glavne dijagonale).

U mnogim metodama biće Vam potrebna tolerancija ε. Kao toleranciju svuda uzimajte isključivo vrijednosti dobijene pozivom funkcije "`GetEpsilon`" nad matricom koja se obrađuje.

Obavezno napišite i testni glavni program u kojem ćete testirati sve elemente napisanih klasa, i to na nekoliko različitih matrica. Trebate obavezno testirati i situacije koje dovode do bacanja izuzetaka (posebno situacije kada su u pitanju singularne matrice). Obavezno testirajte i slučajeve matrica kod kojih je $a_{1,1} = 0$ (bez pivotizacije, Gaussova ili Gauss-Jordanova eliminacija te LU faktorizacija Croutovim algoritmom padaju već na samom početku). Sve rezultate koje dobijete uporedite sa rezultatima koje daje Julia da provjerite da li Vam je sve ispravno (nemojte čekati na autotestove, da ne gubite vrijeme). Pregledanje zadaće će se vršiti kombinacijom autotestova i ručnog pregledanja. Za metodu, funkciju ili operator koji nisu temeljito testirani u glavnom programu smatraće se da nisu urađeni, te se za njih neće dobiti nikakvi bodovi!

Vodite računa da indeksi vektora u C++-u idu od nule a ne od jedinice, zbog čega će se petlje kretati u malo drugačijem opsegu nego u pseudokôdovima prezentiranim na predavanju. Za tu svrhu, najbolje je da u knjizi pročitate dodatak nazvan "Problemi s indeksacijom nizova, vektora i matrica" koji sadrži informacije koje olakšavaju tu transformaciju. Nemojte za realizaciju algoritama koristiti operator "`()`" implementiran u klasama "`Vector`" i "`Matrix`". Naime, time se algoritmi znatno usporavaju (tipično s faktorom 2 ili više), jer upotreba ovog operatora vrši bespotrebnu provjeru validnosti indeksa (pored toga što vrši njihovo pomjeranje sa svrhom prilagođavanja matematičkoj konvenciji), za koje se zna da će tokom izvođenja svih ovih algoritama biti garantirano ispravni.