Zadaća 1

Ova zadaća nosi **2 poena** i predstavlja *pripremnu zadaću* za sljedeću zadaću (tačnije, sljedeća zadaća će biti *nadogradnja* ove zadaće), tako da studenti koji ne urade ovu zadaću neće moći uraditi ni sljedeću. Ova zadaća se, osim jednog sitnog detalja vezanog za mašinski epsilon, praktično u potpunosti oslanja na predznanja iz predmeta "Tehnike programiranja" i "Linearna algebra i geometrija", tj. ne zahtijeva nikakva specijalna znanja stečena na ovom kursu. Rok za predaju je **srijeda**, **8. XI 2017.** 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 sastavite klase "`Vector`" i "`Matrix`" koje će služiti kao podrška za rješavanje elementarnih problema linearne algebre (ove klase će se dodatno nadograđivati u sljedećoj zadaći novim funkcionalnostima). Klasa "`Vector`" (sa velikim početnim "V") je vrlo jednostavna klasa, koja predstavlja omotač oko generičke klase "`vector`" iz istoimene biblioteke, specijalizirane za tip elemenata "**`double`**". Njen jedini atribut je standardni bibliotečki vektor (tj. objekat tipa "`vector`" iz istoimene biblioteke) čiji su elementi tipa "**`double`**", dok joj je interfejs sljedeći:

```
explicit Vector(int n);
Vector(std::initializer_list<double> l);
int NElems() const;
double &operator[](int i);
double operator[](int i) const;
double &operator()(int i);
double operator()(int i) const;
double Norm() const;
friend double VectorNorm(const Vector &v);
double GetEpsilon() const;
void Print(char separator = '\n', double eps = -1) const;
friend void PrintVector(const Vector &v, char separator = '\n',
  double eps = -1);
friend Vector operator +(const Vector &v1, const Vector &v2);
Vector &operator +=(const Vector &v);
friend Vector operator -(const Vector &v1, const Vector &v2);
Vector &operator -=(const Vector &v);
friend Vector operator *(double s, const Vector &v);
friend Vector operator *(const Vector &v, double s);
Vector &operator *=(double s);
friend double operator *(const Vector &v1, const Vector &v2);
friend Vector operator /(const Vector &v, double s);
Vector &operator /=(double s);
```

Konstruktor sa cjelobrojnim parametrom postavlja dimenziju vektora i postavlja mu elemente na nulu, U slučaju da je dimenzija negativna ili 0, treba baciti izuzetak tipa "`range_error`" uz prateći tekst "Bad dimension". Sekvencijski konstruktor inicijalizira elemente vektora na osnovu inicijalizacione liste, što omogućava kreiranje vektora konstrukcijom poput

```
Vector v{3, 4, 2, 7, 1};
```

Pri tome, sekvencijski konstruktor ne smije dozvoliti da se zadavanjem prazne inicijalizacione liste zada vektor dimenzije 0. Ukoliko se to pokuša, treba baciti isti izuzetak kao pri pokušaju zadavanja dimenzije 0 putem klasičnog konstruktora.

Funkcija članica "`NElems`" vraća broj elemenata vektora nad kojim je primijenjena. Operator "`[]`" omogućava pristup elementima vektora na isti način kao sa elementima standardnog tipa "`vector`", tj. sa indeksacijom od nule, i bez kontrole legalnosti indeksa. Treba podržati konstantnu i nekonstantnu verziju. Sličnu ulogu ima i operator "`()`", samo kod njega indeksacija ide od jedinice, a u slučaju indeksa izvan opsega baca se izuzetak tipa "`range_error`" uz prateći tekst "Invalid index".

Funkcija članica "`Norm`" vraća euklidsku normu vektora nad kojim je primijenjena, koja je data kao kvadratni korijen iz sume kvadrata svih elemenata (euklidska norma je poznata i kao apsolutna vrijednost vektora, a za slučaj vektora u dvije ili tri dimenzije to je prosto njegova dužina). Prijateljska funkcija "`VectorNorm`" radi potpuno istu stvar, samo se euklidska norma računa za vektor koji joj je proslijeđen kao parametar.

Funkcija članica "`GetEpsilon`" vraća kao rezutat produkt euklidske norme vektora (koja se može dobiti pozivom funkcije članice "`Norm`") i mašinskog epsilona za tip "**`double`**". Ova funkcija će se često koristiti kao pomoćna funkcija na raznim drugim mjestima, s obzirom da će se ovako dobijena vrijednost često koristiti kao podrazumijevana tolerancija za greške koje nastaju zbog nesavršenosti računarske aritmetike s realnim brojevima.

Funkcija članica "`Print`" ispisuje sve elemente vektora, razdvojene znakom koji se zadaje kao prvi parametar, pri čemu se iza posljednjeg elemenata se taj znak ne ispisuje. Izuzetak je u slučaju kada je taj znak oznaka za prelazak u novi red. U tom slučaju, prelazak u novi red se vrši i nakon posljednjeg elementa. Oznaka za prelazak u novi red je ujedno i podrazumijevana vrijednost ovog parametra, koja se koristi ukoliko se on izostavi. Funkcija članica "`Print`" posjeduje još jedan parametar, koji predstavlja prag tolerancije za ispis. Naime, svi elementi vektora čija je apsolutna vrijednost manja od ovog praga, ispisuju se kao 0 (ovo će kasnije služiti da se pri ispisu "počiste" izvjesne greške do kojih može doći uslijed akumulacije grešaka u računanju). Ukoliko ovaj parametar ima negativnu vrijednost, kao prag se uzima vrijednost dobijena pozivom funkcije "`GetEpsilon`". Kako ovaj parametar ima podrazumijevanu vrijednost -1, upravo će se takav prag podrazumijevati ukoliko se parametar izostavi. Pored opisane funkcije članice "`Print`", treba podržati i prijateljsku funkciju "`PrintVector`" koja radi identičnu stvar kao i funkcija članica "`Print`", uz jedinu razliku što se ne ispisuje vektor nad kojim je funkcija pozvana, nego vektor koji se prenosi ovoj funkciji kao prvi parametar.

U klasi "`Vector`" treba podržati još nekoliko operatora. Operatori "`+`" i "`-`" omogućavaju sabiranje odnosno oduzimanje dva vektora, pod uvjetom da su iste dimenzije. U suprotnom, baca se izuzetak tipa "`domain_error`" uz prateći tekst "Incompatible formats". Operator "`*`" omogućava množenje skalara (realnog broja) sa vektorom, odnosno vektora sa skalarom, te računanje skalarnog proizvoda dva vektora. U ovom posljednjem slučaju, vektori moraju biti iste dimenzije, u suprotnom se baca izuzetak tipa "`domain_error`" uz prateći tekst "Incompatible formats". Operator "`/`" omogućava dijeljenje vektora sa skalarom. U slučaju dijeljenja nulom, treba baciti izuzetak istog tipa uz prateći tekst "Division by zero". Pored ovih operatora, treba podržati su i njihove varijante sa dodjelom "`+=`" i "`-=`","`*=`" i "`/=`".

Klasa "`Matrix`" je ključna klasa za implementaciju algoritama vezanih za rješavanje problema linearne algebre. Njen jedini atribut će biti matrica organizirana kao vektor vektōrâ, gdje pod vektorom smatramo standardni bibliotečki vektor (tj. objekat tipa "`vector`" iz istoimene biblioteke). Elementi matrice trebaju biti tipa "**`double`**", Ova klasa za sada treba imati sljedeći interfejs (ovaj interfejs će se proširivati novim funkcionalnostima u sljedećoj zadaći):

```
Matrix(int m, int n);
Matrix(const Vector &v);
Matrix(std::initializer_list<std::vector<double>> l);
int NRows() const;
int NCols() const;
double *operator[](int i);
const double *operator[](int i) const;
double &operator()(int i, int j);
double operator()(int i, int j) const;
double Norm() const;
friend double MatrixNorm(const Matrix &m);
double GetEpsilon() const;
void Print(int width = 10, double eps = -1) const;
friend void PrintMatrix(const Matrix &m, int width = 10, double eps = -1);
friend Matrix operator +(const Matrix &m1, const Matrix &m2);
Matrix &operator +=(const Matrix &m);
friend Matrix operator -(const Matrix &m1, const Matrix &m2);
Matrix &operator -=(const Matrix &m);
friend Matrix operator *(double s, const Matrix &m);
friend Matrix operator *(const Matrix &m, double s);
Matrix &operator *=(double s);
friend Matrix operator *(const Matrix &m1, const Matrix &m2);
Matrix &operator *=(const Matrix &m);
friend Vector operator *(const Matrix &m, const Vector &v);
friend Matrix Transpose(const Matrix &m);
void Transpose();
```

Konstruktor sa dva cjelobrojna parametra postavlja dimenziju matrice i postavlja joj elemente na nulu. U slučaju da je neka od dimenzija negativna ili 0, treba baciti izuzetak tipa "`range_error`" uz

prateći tekst "Bad dimension". Konstruktor koji prima objekat tipa "`Vector`" omogućava pretvorbu vektora (tj. objekata tipa "`Vector`") u matrice (tj. objekte tipa "`Matrix`") tretirajući vektore kao matrice sa samo jednom kolonom, dok sekvencijski konstruktor inicijalizira elemente matrice na osnovu inicijalizacione liste, što omogućava kreiranje matrice konstrukcijom poput

```
Matrix m{{2, 5, 4, 1}, {3, 2, 6, 5}, {4, 1, 5, 2}};
```

Sekvencijski konstruktor ne smije dozvoliti da se putem njega zada matrica čija je neka od dimenzija 0 (npr. zadavanjem prazne liste, ili liste čiji su elementi prazne liste). Ukoliko se to pokuša, treba baciti isti izuzetak kao da se putem klasičnog konstruktora sa 2 parametra pokuša kreirati takva matrica. Također, sekvencijski konstruktor ne smije dozvoliti da se putem njega zada grbava matrica. Ukoliko se to pokuša, treba baciti izuzetak tipa "`logic_error`" uz prateći tekst "Bad matrix".

Metode "`NRows`" i "`NCols`" vraćaju broj redova odnosno broj kolona matrice. Operator "`[]`" vraća pokazivač na odgovarajući red matrice (uz indeksaciju redova od nule), što dalje omogućava da se uz još jednu indeksaciju pristupa elementima matrice kao da se radi o klasičnom dvodimenzionalnom nizu ili klasičnom vektoru vektōrâ (uz indeksaciju redova i kolona od nule i bez kontrole legalnosti indeksa). Treba podržati konstantnu i nekonstantnu verziju. Operator "`()`" omogućava alternativni pristup elementima matrice uz navođenje indeksa reda i kolone unutar zagrada (razdvojeni zarezom), uz indeksaciju koja ide od jedinice (u skladu sa uobičajenim matematskim konvencijama), pri čemu se u slučaju indeksa izvan opsega baca se izuzetak tipa "`range_error`" uz prateći tekst "Invalid index". Ovdje također treba podržati konstantnu i nekonstantnu verziju.

Funkcija članica "`Norm`" vraća Frobeniusovu normu matrice nad kojomm je primijenjena, koja je data kao kvadratni korijen iz sume kvadrata svih elemenata matrice. Prijateljska funkcija "`MatrixNorm`" radi potpuno istu stvar, samo se Frobeniusova norma računa za matricu koja joj je proslijeđena kao parametar. Slično kao i za slučaj vektora, i ovdje treba podržati funkciju članicu "`GetEpsilon`" koja vraća kao rezutat produkt Frobeniusove norme matrice i mašinskog epsilona za tip "**`double`**".

Funkcija članica "`Print`" ispisuje sve elemente matrice, red po red (uz prelazak u novi red nakon što se ispišu svi elementi jednog reda), pri čemu se za svaki element rezervira prostor čija se širina zadaje kao prvi parametar (koji ima podrazumijevanu vrijednost 10). Slično kao kod funkcije članice "`Print`" u klasi "`Vector`", drugi parametar predstavlja prag tolerancije, tako da se svi elementi matrice koji su manji po apsolutnoj vrijednosti od ovog praga ispisuju kao nule. Ukoliko je ovaj parametar negativn (što je i podrazumijevana vrijednost), kao prag se koristi vrijednost dobijena pozivom funkcije članice "`GetEpsilon`". Treba podržati i prijateljsku funkciju "`PrintMatrix`" koja radi identičnu stvar kao i funkcija članica "`Print`", uz jedinu razliku što se ne ispisuje matricu nad kojom je funkcija pozvana, nego matricu koja se prenosi ovoj funkciji kao prvi parametar.

Operatori "`+`", "`-`" i "`*`" treba da podrže sabiranje odnosno oduzimanje dvije matrice, te množenje matrice skalarom (i skalara matricom), kao i množenje dvije matrice (u matričnom smislu), odnosno množenje matrice vektorom (u tom slučaju vektor se posmatra kao vektor kolona, odnosno matrica sa samo jednom kolonom, dok se kao rezultat ponovo dobija vektor, odnosno objekat tipa "`Vector`"). Sabiranje odnosno oduzimanje matrica definirano je samo ukoliko su obje matrice istog formata, dok je matrično množenje definirano samo ukoliko je broj kolona prve matrice jednak broju redova druge matrice. U slučaju da odgovarajuća operacija nije podržana, treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Incompatible formats". Za sva ova tri operatora treba podržati i odgovarajuće verzije sa dodjelom "`+=`", "`-=`" i "`*=`".

Funkcija "`Transpose`", u verziji kao klasična funkcija, vraća kao rezultat transponiranu matricu matrice koja joj je prenesena kao parametar. S druge strane, funkcija članica "`Transpose`" transponira samu matricu nad kojom je pozvana (dakle, mijenja matricu na koju je pozvana). Ukoliko je matrica kvadratna, ova funkcija članica mora raditi "u mjestu", tj, bez alociranja ikakvih dodatnih matrica, vektora, itd. dok za slučaj matrica koje nisu kvadratne, dozvoljeno je koristiti pomoćne matrice (u tom slučaju, veoma je teško izvršiti transpoziciju u mjestu).

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. 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!