Zadaća 6.

Ova zadaća nosi ukupno 4 poena, pri čemu svaki zadatak nosi po 1 poen. Prva tri zadatka se mogu uraditi na osnovu gradiva sa prvih 13 predavanja i pretpostavljenog predznanja iz predmeta "Osnove računarstva", osim dijelova koji se odnose na datoteke, dok posljednji zadatak zahtijeva poznavanje svih predavanja. Ova zadaća je od vitalnog značaja za razumijevanje kursa kao cjeline i studenti koji ne urade ovu zadaću tretiraće se znatno strožije na usmenom ispitu ukoliko žele upisati visoku ocjenu. Rok za predaju ove zadaće je nedjelja, 22. VI 2014. (do kraja dana) i ne može se produžiti (nemojte pitati može li se produžiti do dana ispita – ne može). Zadaće se predaju putem Zamgera.

Napomena: Nije greška činjenica da sve zadaće zajedno nose više od 20 poena (dodatni poeni su kompenzacioni poeni preko kojih možete nadoknaditi poene koje ste ranije izgubili).

1. U raznim oblastima matematike često se javlja potreba za radom sa matricama. Mada jezik C++ u svojim standardnim bibliotekama ne sadrži nikakve specijalističke tipove podataka za rad sa matricama, takvi tipovi se mogu naći u raznim nestandardnim bibliotekama koje se mogu posebno instalirati. Vaš zadatak je da kreirate generičku klasu "`Matrica`" koja će predstavljati jedan jednostavan tip podataka koji modelira matrice, pri čemu tip njihovih elemenata može biti proizvoljnog tipa. Sa ovakvim tipom podataka treba da budu moguće stvari poput sljedećih:

```
Matrica<double> a(5, 5), b(5, 5);
std::cout << "Unesi elemente matrice A: ";
std::cin >> a;
std::cout << "Unesi elemente matrice B: ";
std::cin >> b;
std::cout << "Matrica A+B glasi:\n";
std::cout << std::setw(10) << a + b;
```

   Klasa treba da ima sljedeće elemente:

   a) Privatni atribut koji predstavlja vektor vektôrā (odnosno koji je tipa "`vector`" čiji su elementi ponovo tipa "`vector`") u kojem se interno čuvaju elementi matrice (ovo će ujedno biti jedini atribut klase, tako da čitava klasa zapravo predstavlja pogodan "omotač" oko klasične realizacije matrica kao vektora čiji su elementi ponovo vektori).
   b) Konstruktor bez parametara, koji kreira praznu "matricu" formata 0×0 (jedina uloga ovog konstruktora je da dozvoli mogućnost kreiranja nizova čiji su elementi matrice).
   c) Konstruktor sa dva parametara, koji kreira matricu zadanog broja redova i kolona, pri čemu se redovi i kolone zadaju kao parametri.
   d) Generički konstruktor koji prima kao parametar također tipa "`Matrica`", ali sa drugačijim tipom elemenata. Ovaj konstrkutor bi trebao da omogući recimo da se matrica čiji su elementi tipa "**`double`**" može inicijalizirati matricom čiji su elementi recimo tipa "**`int`**" (naravno, tako nešto može raditi samo ukoliko je tip elemenata izvorne matrice konvertibilan u tip elemenata odredišne matrice). Ovaj konstruktor će indirektno (putem automatske prevrobe tipa) omogućiti i međusobno dodjeljivanje matrica različitih tipova elemenata. Napomena: konstruktori i funkcije članice također mogu biti generički (to se koristi recimo ukoliko je tip parametara djelimično ili potpuno nepoznat), mada to nije eksplicitno naglašeno na predavanju. U tom slučaju, "**`template`**" deklaraciju treba staviti ispred deklaracije ili definicije konstruktora odnosno funkcije članice, kao i u slučaju običnih generičkih funkcija.
   e) Konstruktor sa koji prima jedan parametar tipa vektora vektôrā, čiji su elementi istog tipa kao što je pretpostavljeni tip elemenata matrice. Ovaj konstruktor kreira matricu na osnovu elemenata vektora vektôrā koji mu je proslijeđen kao parametar. Osnovna namjena ovog konstruktora je da podrži automatsku pretvorbu vektora vektôrā u odgovarajući tip "`Matrica`". Ukoliko zadani vektor vektôrā nema formu matrice (tj. ukoliko mu svi redovi nemaju isti broj elemenata), treba baciti izuzetak tipa "`logic_error`" uz odgovarajući prateći tekst. Napomena: zbog automatske konverzije inicijalizacionih listi u vektore, ovaj konstruktor će automatski podžati konstrukcije poput "`Matrica m<double>({{1, 2}, {3, 4}})`".

f) Sekvencijski konstruktor koji će podržati kreiranje matrica direktno iz inicijalizacijskih listi, tj. konstrukcije poput "`Matrica m<double>{{1, 2}, {3, 4}}`" (primijetite da za razliku od prethodnog konstruktora, ovdje nema dodatnih okruglih zagrada). Slično prethodnom konstruktoru, i ovaj konstruktor baca izuzetak ukoliko nije zadana ispravna forma matrice.

g) Metodu "`PostaviDimenziju`" sa dva parametra, koja omogućava naknadnu promjenu dimenzija matrice (nove dimenzije se zadaju kao parametri).

h) Metode "`DajBrojRedova`" i "`DajBrojKolona`" bez parametara, koja vraćaju kao rezultat broj redova odnosno broj kolona matrice.

i) Preklopljen unarni operator "`!`", koji primijenjen na matricu daje "**`true`**" ukoliko je matrica nul-matrica (tj. ukoliko su joj svi elementi nule), a "**`false`**" u suprotnom.

j) Metodu "`Transponiraj`" bez parametara koja vrši transpoziciju matrice na koju je primijenjena (vodite računa da matrica ne mora biti kvadratna). U slučaju da je matrica kvadratna, transpoziciju obavite *bez upotrebe pomoćnih matrica* (u slučaju kad matrica nije kvadratna, to je znatno teže, mada ne i nemoguće izvesti).

k) Klasičnu funkciju (ne članicu) "`Transpozicija`" sa jednim parametrom koja kao rezultat daje transponovanu matricu matrice koja joj je data kao parametar (pri tome, ta matrica ostaje neizmijenjena).

l) Preklopljene binarne operatore "`+`", "`-`" i "`*`" koji redom nalaze zbir, razliku i proizvod matrica, pod uvjetom da su matrice međusobno saglasne za izvođenje tih operacija. U suprotnom, treba baciti izuzetak tipa "`domain_error`" uz odgovarajući prateći tekst. Matrice ne moraju biti istog tipa elemenata (recimo, može se sabrati matrica čiji su elementi tipa "**`int`**" sa elementima tipa "**`double`**", pri čemu rezultirajuća matrica treba imati onakav tip elemenata kakav se dobije primjenom odgovarajuće operacije nad elementima jedne i druge matrice (uputa: ovdje će Vam trebati "**`decltype`**" operator).. Operator "`*`" također treba da podržava množenje broja sa matricom odnosno matrice sa brojem. Tip broja ne mora biti isti kao tip elemenata matrice. Odgovarajuće operatorske funkcije obavezno treba implementirati *izvan deklaracije klase* (upadnete li u probleme, informacije date pred kraj Predavanja 11_b mogu Vam biti od koristi).

m) Prekopljene binarne operatore "`+=`","`*=`" i "`-=`" takve da izrazi poput "`X += Y`", "`X -= Y`" i "`X *= Y`" uvijek imaju isti efekat kao i izrazi "`X = X + Y`", "`X = X - Y`" i "`X = X * Y`" kad god to ima smisla. Pri tome, realizacije ovih operatora se ne smiju trivijalno svoditi na operatore "`+`","`-`" i "`*`" (tj. nije dozvoljena implementacija operatora "`+=`" koja se prosto sastoji od naredbe poput "**`return`** `x = x + y`").

n) Preklopljene binarne operatore "`==`" i "`!=`" koji daju rezultat "**`true`**" ukoliko su operandi jednake odnosno različite matrice, i "**`false`**" u suprotnom.

o) Preklopljen operator "`()`" koji omogućava pristup elementima matrice zadavanjem reda i kolone kao parametara u zagradi, npr. "`m(2, 3)`" je element u drugom redu i trećoj koloni. Indeksi se gledaju "matematički", dakle od jedinice a ne od nule. Ukoliko indeksi nisu u dozvoljenom opsegu, treba baciti izuzetak tipa "`range_error`". Rezultat treba biti referenca na element, tako da se može koristiti i sa lijeve strane operatora dodjele, osim u slučaju kada se ovaj operator primijeni na konstantnu matricu (tada umjesto reference treba vratiti kopiju elementa).

p) Preklopljen operator "`[]`" koji omogućava pristup elementima matrice sintaksom kao da se radi o klasičnim dvodimenzionalnim nizovima (npr. "`m[5][2]`", sa indeksacijom *od nule* i *bez provjere* ispravnosti indeksa. Potrebno je podržati konzistentan tretman konstantnih matrica (tj. da se za slučaj konstantnih matrica rezultat ne može koristiti sa lijeve strane operatora dodjele). Uputa: dovoljno je vratiti *referencu* na odgovarajući red matrice, koji je po tipu vektor, nakon ćega će druga indeksacija automatski odraditi svoj posao, jer se primjenjuje na vektor. Za slučaj konstantnih matrica, umjesto da se vraća kopija čitavog reda, efikasnije je vratiti konstantnu referencu na odgovarajući red.

r) Preklopljen operator "`<<`" koji treba da podrži ispis matrice na izlazni tok. Elementi matrice se ispisuju red po red (elementi jednog reda matrice u jednom redu, a nakon svakog reda matrice prelazi se u novi red). Svaki element treba zauzeti onoliko prostora koliko je postavljeno manipulatorom "`setw`" ili pozivom funkcije "`width`" nad objektom toka. Uputa: da biste unutar operatorske funkcije saznali kolika je trenutna postavka širine, pozovite funkciju

"`width`" nad objektom toka, ali *bez parametara*. Ono što dobijete kao rezultat je trenutna postavka širine.

s) Preklopljen operator "`>>`" koji treba da podrži unos matrice sa ulaznog toka. Matrica se unosi prosto kao slijed elemenata matrice koji su razmaknuti bjelinom (razmakom, tabulatorom ili prelaskom u novi reda).

Primijetimo da destruktor, kopirajući i pomjerajući konstruktori te preklopljeni operatori dodjele neće biti potrebni za ispravno funkcioniranje klase, s obzirom da se svi elementi matrice čuvaju u vektoru vektôrā, koji se brine za automatsko upravljanje memorijom.

Sve metode implementirajte izvan deklaracije klase, osim trivijalnih metoda koje trebate implementirati direktno unutar deklaracije klase. Metode koje su inspektori treba deklarirati kao takve. Obavezno napišite i testni program u kojem ćete demonstrirati sve elemente napisane generičke klase.

2. Za potrebe nekog fakulteta neophodno je vršiti evidenciju o polaznicima (studentima) i njihovom uspjehu. Za tu svrhu fakultet koristi računarski program u kojem je definirane i implementirane apstraktna bazna klasa nazvana "`Student`", zatim dvije konkretne klase "`StudentBachelor`" i "`StudentMaster`" izvedene iz bazne klase "`Student`", kao i klasa "`Fakultet`" koja predstavlja kolekciju objekata izvedenih iz klase "`Student`". Bazna klasa "`Student`" treba da ima sljedeće elemente:

   a) Privatne atribute koji čuvaju podatke o imenu i prezimenu studenta (tipa string), broju indeksa (cijeli broj), broju položenih ispita (cijeli broj) i prosječnoj ocjeni (realan broj), i nikave druge atribute.
   b) Konstruktor čiji su parametri ime i prezime studenta, kao i broj indeksa studenta, koji odgovarajuće atribute inicijalizira na vrijednosti zadane parametrima, dok se broj položenih ispita i prosječna ocjena inicijaliziraju respektivno na 0 i 5.
   c) Trivijalne metode "`DajIme`", "`DajPrezime`", "`DajBrojIndeka`", "`DajBrojPolozenih`" i "`DajProsjek`" koje vraćaju vrijednosti svih odgovarajućih atributa.
   d) Metodu "`RegistrirajIspit`" sa jednim parametrom, koja služi za registraciju novog ispita, a koja povećava broj položenih ispita za jedinicu i ažurira prosječnu ocjenu u skladu sa ocjenom iz novopoloženog ispita, koja se zadaje kao parametar (u slučaju neispravne ocjene, treba baciti izuzetak tipa "`domain_error`"). Ocjena 5 se prihvata kao legalna, ali se ignorira (tj. niti se ažurira prosjek, niti se povežava broj položenih ispita).
   e) Metodu "`PonistiOcjene`" bez parametara, koja poništava sve registrirane ocjene (tj. dovodi objekat u stanje kakvo je bilo odmah nakon kreiranja objekta).
   f) Apstraktnu virtualnu metodu "`IspisiPodatke`" bez parametara.
   g) Apstraktnu metodu "`DajKopiju`" bez parametara. Ova metoda će se koristiti za potrebe konstruktora kopije klase "`Fakultet`", a njene implementacije u izvedenim klasama treba da obezbijede kreiranje identične kopije objekta nad kojim su pozvane, uz vraćanje adrese novokreirane kopije kao rezultata.

   Izvedena klasa "`StudentBachelor`" razlikuje se od bazne klase "`Student`" samo po tome što sadrži konkretnu implementaciju apstraktnih metoda "`IspisiPodatke`" i "`DajKopiju`". Metoda "`DajKopiju`" treba da kreira identičnu kopiju objekta tipa "`StudentBachelor`" nad kojim je pozvana, dok metoda "`IspisiPodatke`" u ovoj klasi treba ispisati podatke o studentu u obliku

   Student bachelor studija <***ime***> <***prezime***>, ***sa brojem indeksa*** <***indeks***>,
   ima prosjek <***prosjek***>**.**

   Izvedena klasa "`StudentMaster`" razlikuje se od bazne klase "`Student`" prvo po tome što sadrži dodatni atribut koji čuva informaciju kada je student završio prvi stepen studija (bachelor studij), kao i dodatni parametar u konstruktoru koji omogućava postavljanje ovog atributa. Naravno, metodu "`DajKopiju`" u ovoj klasi treba implementirati tako da kreira identičnu kopiju objekta tipa "`StudentMaster`" nad kojim je pozvana, dok implementacija metode "`IspisiPodatke`" u ovoj klasi treba da ispisuje podatke o studentu u sljedećem obliku:

   Student master studija <***ime***> <***prezime***> ***sa brojem indeksa*** <***indeks***>**,**
   završio bachelor studij godine <***godina***>**, *ima prosjek*** <***prosjek***>**.**

Klasa "`Fakultet`" treba da ima sljedeće elemente:

a) Privatni atribut koji predstavlja vektor pokazivača na objekte negog od tipova izvedenih iz tipa "`Student`" koji sadrže podatke u upisanim studentima.
b) Konstruktor bez parametara, koji omogućava kreiranje objekata tipa "`Fakultet`" (bez ikakvih dodatnih informacija).
c) Kopirajući konstruktor i kopirajući operator dodjele koji omogućavaju da se objekti tipa "`Fakultet`" mogu sigurno kopirati i međusobno dodjeljivati korištenjem strategije dubokog kopiranja Za potrebe kreiranja kopija individualnih objekata pohranjenih u kolekciji koristite funkciju "`DajKopiju`".
d) Pomjerajući konstruktor i pomjerajući operator dodjele koji optimiziraju postupak kopiranja za slučaj kada su u igri privremeni objekti.
e) Destruktor koji oslobađa sve resurse koje je objekat tipa "`Fakultet`" zauzeo tokom svog života.
f) Dvije metode istog imena "`UpisiStudenta`" koje služe za upis novog studenta, od kojih jedna ima tri, a druga četiri parametra. Metoda sa tri parametra upisuje studenta bachelor studija, pri čemu su parametri broj indeksa, ime i prezime. Metoda sa četiri parametra upisuje studenta master studija, pri čemu četvrti dodatni parametar predstavlja godinu kada je student završio bachelor studij.
g) Metodu "`RegistirajOcjenu`" koja vrši registraciju nove ocjene. Metoda kao parametre zahtijeva broj indeksa studenta kao i ocjenu koju je student dobio. Ova metoda treba da ažurira prosječnu ocjenu studenta na osnovu novounesene ocjene. U slučaju da student sa zadanim brojem indeksa ne postoji, treba baciti izuzetak tipa "`domain_error`".
h) Metodu "`ObrisiStudenta`" koja briše studenta sa brojem indeksa koji se zadaje kao parametar. U slučaju da student sa zadanim brojem indeksa ne postoji, treba baciti izuzetak tipa "`domain_error`".
i) Metodu "`PonistiSveOcjene`" koja poništava sve podatke o unesenim ocjenama za studenta sa brojem indeksa koji se zadaje kao parametar (tj. broj položenih ispita se vraća na nulu, a prosjek na 5). U slučaju da student sa zadanim brojem indeksa ne postoji, treba baciti izuzetak tipa "`domain_error`".
j) Metodu "`IspisiPodatkeOStudentu`" koja ispisuje podatke o studentu sa brojem indeksa koji se zadaje kao parametar. U slučaju da student sa zadanim brojem indeksa ne postoji, treba baciti izuzetak tipa "`domain_error`".
k) Metodu "`IspisiStudenteSaProsjekomVecimOd`" koja ispisuje imena i prezimena svih studenata čija je prosječna ocjena veća od vrijednosti koja se zadaje kao parametar, ili informaciju da takvih studenata nema.
l) Metodu "`IspisiStudenteKojiSuPoloziliViseOd`" koja ispisuje imena i prezimena svih studenata koji su položili više ispita od broja koji se zadaje kao parametar, ili informaciju da takvih studenata nema.
m) Metodu "`IspisiSveStudente`" koja ispisuje spisak svih studenata sa pripadnim podacima (pozivom metode "`IspisiPodatke`" za svakog studenta), sortiran u opadajućem poretku po prosječnoj ocjeni (tj. student sa najvećom prosječnom ocjenom se ispisuje prvi).
n) Metodu "`UcitajIzDatoteke`" koja čita podatke o studentima i ocjenama iz tekstualnih datoteka. Ova metoda ima dva parametra (koji su po tipu konstantni nizovi znakova) koji redom predstavljaju imena datoteka sa podacima o studentima, te podacima o ocjenama. Datoteka o studentima u svakom redu sadrži podatke za po jednog studenta, i to redom ime, prezime, broj indeksa i godinu završetka bachelor studija (ili 0 ako se radi o studentu bachelor studija). Na primjer:

```
Meho Mehić 11 0
Ibro Ibrić 12 1998
Vaso Vasić 13 0
Ivo Ivić 14 0
Marko Marković 15 2001
```

Datoteka sa podacima o ocjenama u svakom redu sadrži prvo broj indeksa studenta čija se ocjena upisuje, a zatim samu ocjenu. Na primjer:

```
11 8
11 7
11 9
12 7
12 8
13 6
13 19
14 7
14 8
14 8
15 9
```

Ukoliko je prilikom poziva ove metode već bilo unesenih podataka o studentima, stari podaci prethodno trebaju biti izbrisani i zamijenjeni novoučitanim podacima. Recimo, nakon unosa podataka iz gore navedenih datoteka, sortirani spisak studenata trebao bi izgledati ovako:

Student master studija Marko Marković, sa brojem indeksa 15,
završio bachelor studij godine 2001, ima prosjek 9.
Student bachelor studija Meho Mehić, sa brojem indeksa 11,
ima prosjek 8.
Student bachelor studija Vaso Vasić, sa brojem indeksa 13,
ima prosjek 8.
Student bachelor studija Ivo Ivić, sa brojem indeksa 14,
ima prosjek 7.66.
Student master studija Ibro Ibrić sa brojem indeksa 12,
završio bachelor studij godine 1998, ima prosjek 7.5.

U slučaju da dođe do bilo kakvih problema prilikom čitanja datoteka (uključujući i slučajeve kada ulazne datoteke sadrže podatke koji nisu u skladu sa očekivanjima), treba baciti izuzetak tipa "`logic_error`".

Napisane klase testirajte u programu koji će testirati sve njihove elemente.

3. Neka je *vozilo* objekat koji je, između ostalog, karakteriziran svojom težinom. *Automobil* je specijalna vrsta vozila, koje može primiti određeni manji broj putnika od kojih svaki ima svoju težinu. *Kamion* je specijalna vrsta vozila koje se može nakrcati teretom određene težine. *Autobus* je sličan automobilu, ali je predviđen za veći broj putnika.

   a) Razvijte hijerarhiju klasa koje opisuju ove objekte. Bazna klasa "`ApstraktnoVozilo`" predstavlja apstraktnu baznu klasu koja sadrži ono što je zajedničko za sve razmatrane vrste vozila. Ona posjeduje atribut koji predstavlja težinu vozila (tipa cijeli broj), konstruktor koji inicijalizira ovaj atribut, te četiri metode nazvane "`DajTezinu`", "`DajUkupnuTezinu`", "`DajKopiju`" i "`IspisiPodatke`". Prva metoda daje vlastitu težinu vozila.. Druga metoda je u baznoj klasi apstraktna i predviđena je da daje ukupnu težinu vozila u koju je uračunata i težina svega što se u vozilu nalazi. Treća metoda je također apstraktna, a predviđena je da kreira kopiju objekta nad kojim je pozvana i da vrati kao rezultat adresu kreirane kopije. Ova metoda će služiti za potrebe polimorfnog kopiranja. Konačno, i metoda "`IspisiPodatke`" je apstraktna, a namijenjena je za ispis podataka o tipu vozila, te njegovoj vlastitoj i ukupnoj težini (stil ispisa oblikujte po vlastitoj volji). Naravno, sve apstraktne metode moraju biti izvedene tako da ukoliko se svim opisanim objektima pristupa preko pokazivača na baznu klasu "`ApstraktnoVozilo`", uvijek treba da bude pozvana ispravna verzija metode, koja će uzeti u obzir specifičnosti objekta.

   Klasa "`Automobil`" nasljeđuje se iz klase "`ApstraktnoVozilo`", a posjeduje dodatni atribut koji je tipa vektor cijelih brojeva, koji čuva težine putnika u vozilu. Konstruktor ove klase ima dodatni parametar, koji predstavlja vektor težina putnika. Zbog činjenice da postoji automatska konverzija inicijalizacionih listi u vektore, automatski će biti moguće konstrukcije

tipa "`Automobil a(700, {80, 90, 60})`". Također, ova klasa sadrži konkretne realizacije metoda "`DajUkupnuTezinu`", "`DajKopiju`" i "`IspisiPodatke`".

Klasa "`Kamion`" se također nasljeđuje iz klase "`ApstraktnoVozilo`", a posjeduje dodatni cjelobrojni atribut koji predstavlja težinu tereta. Ova klasa također ima dodatni parametar u konstruktoru (težina tereta), te konkretne realizacije svih apstraktnih metoda.

Klasa "`Autobus`" slična je klasi "`Automobil`", tako da se i ona također nasljeđuje iz klase "`ApstraktnoVozilo`". Kako autobus može prevoziti mnogo putnika, ne čuva se informacija o težini svakog od njih, nego se čuva informacija o broju putnika i njihovoj prosječnoj težini (tako da je ukupna težina svih putnika jednaka proizvodu broja putnika i prosječne težine). Konstruktor ove klase ima dva dodatna parametra u odnosu na baznu klasu (broj putnika i prosječna težina jednog putnika), te konkretne realizacije svih apstraktnih metoda.

b) Razvijte surogatsku klasu "`Vozilo`" koja predstavlja polimorfni omotač za proizvoljnu vrstu vozila. Promjenljive tipa "`Vozilo`" moraju biti takve da se u njih može smjestiti bilo automobil, bilo kamion, bilo autobus (tj. sadržaj promjenljive tipa "`Automobil`", "`Kamion`" ili "`Autobus`") odnosno promjenljiva bilo kojeg tipa koji je izveden iz apstraktnog tipa "`ApstraktnoVozilo`" (što uključuje i tipove koji će eventualno biti kreirani u budućnosti). Naravno, sa promjenljivim tipa "`Vozilo`" mogu se raditi sve operacije koje se mogu raditi sa bilo kojom vrstom vozila (takvih operacija ovdje nema mnogo, ali to je samo da zadatak ne bude dugačak), mogu se bezbjedno kopirati, međusobno dodjeljivati, itd.

c) Napišite testni program koji čita podatke o vozilima iz tekstualne datoteke u vektor čiji su elementi vozila (tj. koji su tipa "`Vozilo`"), a zatim sortira vozila po ukupnoj težini u rastući poredak (koristeći funkciju "`sort`") i na kraju ispisuje ukupne težine vozila nakon sortiranja (svaka težina u posebnom redu). Svaki red datoteke sadrži podatke o jednom vozilu. Za slučaj automobila, prvi znak u redu je "`A`", nakon čega slijedi vlastita težina automobila, broj putnika i težina svakog od putnika, na primjer "`A500 3 80 60 75`" za automobil težine 500 sa 3 putnika težina 80, 60 i 75 respektivno. Za slučaj kamiona, prvi znak u redu je "`K`", nakon čega slijedi težina kamiona i težina tereta, na primjer "`K500 1200`" za kamion težine 500 natovaren sa teretom težine 1200. Konačno, za slučaj autobusa, prvi znak u redu je "`B`", nakon čega slijedi težina autobusa, broj putnika i prosječna težina putnika, na primjer "`B1500 50 80`" za autobus težine 1500 sa 50 putnika čija je prosječna težina 80. U slučaju bilo kakvih problema pri čitanju datoteke (što uključuje i slučaj kada datoteka sadrži neispravne podatke) treba ispisati prikladne poruke o greški.

4. Na Predavanju 14_b razmotrena su dva primjera programa od kojih prvi kreira binarnu datoteku koja sadrži podatke o studentima nazvanu "`STUDENTI.DAT`", dok drugi program iščitava sadržaj kreirane datoteke i obračunava prosjek studenata na osnovu podataka u datoteci. Dopunite ovu kolekciju programa trećim programom koji sortira sadržaj datoteke "`STUDENTI.DAT`" u opadajući poredak prema prosječnoj ocjeni, *bez prethodnog učitavanja datoteke u niz* (dakle, sve manipulacije treba raditi direktno nad datotekom). Za sortiranje koristite neki od naivnih algoritama sortiranja (npr. BubbleSort). U slučaju da dva studenta imaju isti prosjek, prije treba doći onaj čije ime dolazi prije po abecednom poretku. Neocijenjene studente tretirajte kao da imaju prosjek 0.

SAVJET: Nakon što prvim programom kreirate testnu datoteku "`STUDENTI.DAT`", obavezno napravite njenu rezervnu kopiju. Naime, dok Vam program ne proradi, vjerovatno ćete više puta uništiti sadržaj datoteke. Stoga, ukoliko ste napravili rezervnu kopiju, nećete morati svaki put kada uništite sadržaj datoteke pokretati ponovo program za kreiranje datoteke i unositi iznova podatke (što može biti veoma mučno). Umjesto toga, dovoljno je da kopirate rezervnu kopiju u datoteku "`STUDENTI.DAT`".