Zadaća 6.

Ova zadaća nosi ukupno 4,5 poena, pri čemu prvi zadatak nosi 1,3 poena, drugi zadatak 0,8 poena, a treći i četvrti zadatak po 1,2 poena. Zadaci traže poznavanje svih 14 predavanja i pretpostavljeno predznanje iz predmeta "Osnove računarstva", osim Zadatka 2 i većeg dijela Zadatka 1 koji se mogu uraditi i bez Predavanja 14. Rok za predaju ove zadaće je subota, 24. VI 2017. do 9.00.

1. Za vođenje evidencije podataka o robi u nekom skladištu potrebno je razviti kontejnersku klasu nazvanu "`Skladiste`". U skladištu se roba nalazi pohranjena u sanducima (za čvrste predmete), u vrećama (za praškaste materije) i u buradima (za tečnosti). Sanduci, vreće i burad se modeliraju redom pomoću klasa "`Sanduk`", "`Vreca`" odnosno "`Bure`". Sanduk je opisan svojom težinom u kilogramima (realni broj), nazivom predmeta koji se u njemu čuvaju (tipa "`string`"), pri čemu se pretpostavlja se da jedan sanduk čuva samo istovrsne predmete), te težinama predmeta koji se u njemu čuvaju, također u kilogramima (ovi podaci se čuvaju u vektoru realnih brojeva, pri čemu svaki od elemenata vektora odgovara jednom predmetu). Vreća je također opisana svojom težinom (u kilogramima), nazivom praškaste materije koji se u njoj čuva, te težinom pohranjene praškaste materije. Bure je opisano svojom težinom, nazivom tečnosti koja se u njemu čuva, specifičnom težinom (gustinom) tečnosti u kilogramima po metru kubnom (realan broj), te zapreminom pohranjene tečnosti koja se u njemu čuva. Informacijama o robi pohranjenoj u skladištu pristupa se pomoću vektora pametnih pokazivača koji pokazuju na objekte tipa "`Sanduk`", "`Vreca`" ili "`Bure`" (za tu svrhu, sve ove klase moraju biti izvedene iz neke apstraktne bazne klase, koju ćete nazvati "`Spremnik`"). Tom nizu pokazivača se pristupa preko nekog od atributa pohranjenog unutar klase "`Skladiste`". Konstruktor klase "`Sanduk`" kao parametre zahtijeva težinu, naziv predmeta koji se čuvaju, te vektor težina predmeta koji se u njemu čuvaju. Konstruktor klase "`Vreca`" prima kao parametre težinu vreće, naziv pohranjene materije, te težinu pohranjene materije, dok konstruktor klase "`Bure`" prima kao parametre težinu, naziv tečnosti koja se čuva, specifičnu težinu tečnosti, te zapreminu pohranjene tečnosti. Atribute koji su zajednički za sanduke, vreće i burad (težina i naziv pohranjenog sadržaja) treba čuvati u baznoj klasi, dok izvedene klase čuvaju samo atribute specifične za datu vrstu objekta. U baznoj klasi također trebaju biti i metode "`DajTezinu`", "`DajUkupnuTezinu`" i "`Ispisi`" bez parmetara. Prva metoda daje težinu vlastitu težinu sanduka, vreće ili bureta (bez onoga što je u njima), druga radi istu stvar, samo uračunava u obzir i težinu onoga što se nalazi u sanduku ili buretu, dok metoda "`Ispisi`" ispisuje podatke o sanduku ili buretu u obliku poput sljedećeg:

```
Vrsta spremnika: Sanduk
Sadrzaj: Trofazni kataklingeri za auspuhe
Tezine predmeta: 2 3 1 2 2 4 3 1 3 (kg)
Vlastita tezina: 10 (kg)
Ukupna tezina: 31 (kg)

Vrsta spremnika: Vreca
Sadrzaj: Praskaste cincozne za glajfanje
Vlastita tezina: 0.2 (kg)
Tezina pohranjene materije: 5 (kg)
Ukupna tezina: 5.2 (kg)

Vrsta spremnika: Bure
Sadrzaj: Rafinirana kalamuta iz Katange
Vlastita tezina: 5 (kg)
Specificna tezina tecnosti: 1300 (kg/m^3)
Zapremina tecnosti: 150 (l)
Ukupna tezina: 200 (kg)
```

   Razumije se da će metode "`DajUkupnuTezinu`" i "`Ispisi`" morati biti apstraktne, s obzirom da u baznoj klasi nema dovoljno informacija koje omogućavaju njihovu implementaciju

   Što se tiče klase "`Skladiste`", primjerci ove klase treba da se mogu kreirati ne navodeći nikakve dodatne specifikacije. Pri tome, potrebno je podržati da se prilikom kopiranja ili međusobnog dodjeljivanja primjeraka ove klase kreiraju potpune (duboke) a ne plitke kopije. Ovo kopiranje mora da korektno radi i ukoliko se u budućnosti pojavi neki novi tip spremnika osim sanduka, vreća i buradi (s obzirom da se radi o polimorfnoj kolekciji objekata, za tu svrhu će biti potrebno dodati odgovarajuću podršku i u ostale klase).

Za manipulaciju sa podacima u skladištu predviđeno je nekoliko metoda. Na prvom mjestu, metode "`DodajSanduk`", "`DodajVrecu`" odnosno "`DodajBure`" kreiraju novi objekat tipa "`Sanduk`", "`Vreca`" odnosno "`Bure`" i pohranjuju ga u skladište. Parametri ovih metoda isti su kao i parametri konstruktora objekata tipa "`Sanduk`", "`Vreca`" odnosno "`Bure`". Sve ove tri metode kao rezultat vraćaju adresu novokreiranog objekta u formi (običnog) pokazivača čiji je statički tip pokazivač na "`Spremnik`" (ova povratna vrijednost će se obično ignorirati, ali vidjećemo uskoro gdje ova povratna vrijednost može biti bitna). Predviđena je i univerznalna metoda za dodavanje sa dva parametra nazvana "`DodajSpremnik`", koja može dodavati objekte bilo kojeg tipa izvedenog iz bazne klase "`Spremnik`". Prvi parametar je pokazivač na objekat koji želimo dodati, dok je drugi parametar logička vrijednost koja određuje da li se objekat predaje u vlasništvo klasi ili ne. Ukoliko ovaj parametar ima vrijednost "**`true`**", pretpostavlja se da je onaj ko poziva ovu metodu kreirao dinamički ovaj objekat i ne želi se više brinuti o njemu, nego brigu o njemu prepušta kontejnerskoj klasi (tačnije, pametnim pokazivačima koji se u njoj nalaze). S druge strane, ukoliko ovaj parametar ima vrijednost "**`false`**", objekat čiju adresu šaljemo u funkciju ne mora uopće biti dinamički alociran, a i ukoliko jeste, o njegovom brisanju se treba brinuti onaj ko je pozvao funkciju. U tom slučaju (s obzirom da kolekcija svakako mora čuvati dinamički alocirane objekte), treba kreirati dinamičku kopiju tog objekta koja se pohranjuje u kolekciju umjesto samog objekta. Funkcija kao rezultat vraća adresu pohranjenog objekta, što ovisno o slučaju može biti ista adresa koja je poslana kao parametar, ili adresa kreirane dinamičke kopije. Konačno, treba podržati i metodu "`BrisiSpremnik`" koja omogućava brisanje nekog od objekata koji su već pohranjeni u skladištu. Parametar ove funkcije je adresa objekta koji se briše, i to je zapravo ista adresa koju vraćaju funkcije poput "`DodajSanduk`". Slijedi da ako želimo imati mogućnost da brišemo neki objekat (npr. sanduk), moramo negdje sačuvati adresu koja je vraćena prilikom kreiranja objekta, jer je taj objekat moguće izbrisati jedino navodeći tu adresu (s obzirom da ne postoji nikakav drugi identifikator koji bi mogao jednoznačno identificirati taj objekat).

Metode "`DajNajlaksi`" odnosno "`DajNajtezi`" vraćaju reference na najlakši odnosno najteži objekat (sanduk, vreća ili bure) u skladištu, ne računajući ono što je pohranjeno u tom objektu. Ove metode se ne smiju moći pozvati nad konstantnim objektima tipa "`Skladiste`". U slučaju da je skladište prazno, treba baciti izuzetak tipa "`range_error`" uz prateći tekst "Skladiste je prazno". Metoda "`BrojPreteskih`" vraća broj objekata u skladištu čija je ukupna težina (tj. vlastita težina zajedno sa težinom onoga što se u njima nalazi) veća od iznosa koji se zadaje putem cjelobrojnog parametra. Ova metoda se mora moći pozvati i nad konstantnim objektom. Predviđena je i metoda "`IzlistajSkladiste`" koja ispisuje spisak svega što se nalazi u skladištu, sortiran u opadajući poredak po ukupnoj težini. Ispis se vrši pozivom metode "`Ispisi`", bez ikakvih praznih redova između objekata. Metoda se mora moći pozvati i nad konstantnim objektom.

Konačno, treba implementirati i metodu "`UcitajIzDatoteke`" koja čita podatke o robi iz tekstualne datoteke čije se ime zadaje kao parametar i smješta robu u skladište (u slučaju da u skladištu već ima unesene robe, postojeći podaci se brišu). Svaki objekat opisan je sa dva reda u datoteci. U prvom redu se nalazi početno slovo "`S`", "`V`" ili "`B`" (za sanduk, vreću odnosno za bure) iza kojeg nakon jednog razmaka slijedi naziv predmeta, praškaste materije ili tečnosti koje su pohranjene u sanduku, vreći odnosno buretu (npr. "`S Tepsije`", "`V Brasno`" ili "`B Suncokretovo ulje`"). U drugom redu se za slučaj sanduka nalazi težina sanduka, broj predmeta i težina svakog od njih, razdvojeno po jednim razmakom (npr. "`10 9 2 3 1 2 2 4 3 1 3`"), za slučaj vreće tu su težina vreće i težina pohranjene materije (npr. "`0.2 5`") dok se za slučaj bureta nalazi težina bureta, te specifična težina i zapremina tečnosti (npr. "`5 1300 150`"). Ukoliko tražena datoteka ne postoji, treba baciti izuzetak tipa "`logic_error`" uz prateći tekst "Trazena datoteka ne postoji". Isti izuzetak, ali uz prateći tekst "Datoteka sadrzi besmislene podatke" treba baciti ukoliko podaci u datoteci nisu u skladu sa specifikacijama. U slučaju bilo kakvih drugih problema pri čitanju (osim pokušaja čitanja iza kraja datoteke), treba također baciti isti izuzetak, uz prateći tekst "Problemi pri citanju datoteke".

Razvijene klase demonstrirajte u testnom programu koji iščitava podatke iz tekstualne datoteke sa imenom "`ROBA.TXT`", a nakon toga ispisuje spisak svega što se nalazi u skladištu, sortiran u opadajući poredak po ukupnoj težini. U testnom programu obavezno predvidite i hvatanje eventualno bačenih izuzetaka koji se mogu pojaviti.

2. Implementirajte surogatsku klasu "PolimorfniSpremnik" koja predstavlja polimorfni omotač za proizvoljnu vrstu spremnika iz prethodnog zadatka. Preciznije, promjenljive ovog tipa moraju biti takve da se u njih može smjestiti bilo sanduk, bilo vreća, bilo bure (tj. sadržaj promjenjive čiji je tip bilo tip "Sanduk", bilo tip "Vreca", bilo tip "Bure"), pa čak i promjenljiva bilo kojeg tipa koji je izveden iz apstraktnog tipa "Spremnik" (što uključuje i tipove koji će eventualno biti kreirani u budućnosti). Naravno, sa promjenljivim tipa "PolimorfniSpremnik" mogu se raditi sve operacije koje su predviđene da rade sa svim vrstama spremnika neovisno od njihovog podtipa (tj. sve operacije koje su definirane u interfejsu apstraktne bazne klase "Spremnik"), mogu se bezbjedno kopirati, međusobno dodjeljivati, itd. Na primjer:

```
PolimorfniSpremnik s1(Bure(5,"Benzin", 930, 70));     // s1 je bure
PolimorfniSpremnik s2, s3;                            // s2 i s3 su nespecificirani
s2 = Sanduk(3, "Tepsije", {0.5, 0.8, 0.6, 0.5});      // s2 je sada sanduk
s2 = StudentMaster("Paja", "Patak", 4312, 2015);      // a s3 vreca
std::cout << s1.DajTezinu() << std::endl;
std::cout << s2.DajUkupnuTezinu() << std::endl;
s3.Ispisi();
s1 = s2;                                              // sad je i s1 sanduk...
s1.Ispisi();
```

Ukoliko se kreira objekat tipa "PolimorfniSpremnik" bez ikakvih dodatnih informacija, on je na početku nespecificiran i svaki pokušaj poziva bilo koje od metoda nad takvim objektom treba baciti izuzetak tipa "logic_error" uz prateći tekst "Nespecificiran spremnik". Obavezno napišite i testni program u kojem ćete testirati funkcionalnost razvijene surogatske klase.

3. Dopunite generičku klasu "Matrica" koju ste razvili u Zadatku 6 sa Tutorijala 12 (a koja se oslanja na klasu razvijenu na Predavanju 11_b) sa četiri nove metode "SacuvajUTekstualnuDatoteku", "SacuvajUBinarnuDatoteku", "ObnoviIzTekstualneDatoteke" i "ObnoviIzBinarneDatoteke", te jednim dodatnim konstruktorom, koji će kasnije biti opisan (ovo je ujedno dobra prilika da dovršite Zadatak 6 sa Tutorijala 12 ukoliko to već niste uradili prije). Sve ove metode primaju kao parametar naziv datoteke. Metoda "SacuvajUTekstualnuDatoteku" snima sadržaj matrice nad kojom je pozvana u tekstualnu datoteku čije je ime zadano. Datoteka treba formatirana tako da se podaci o svakom redu matrice čuvaju u posebnim redovima datoteke, pri čemu su elementi unutar jednog reda međusobno razdvojeni zarezima (iza posljednjeg reda nema zareza). Recimo, za neku matricu formata 3×4 kreirana datoteka može izgledati poput sljedeće:

```
2.5,-3,1.12,4
0,0.25,3.16,42.3
-1.7,2.5,0,5
```

U slučaju da dođe do bilo kakvih problema pri upisu, treba baciti izuzetak tipa "logic_error" uz prateći tekst "Problemi sa upisom u datoteku". Metoda "SacuvajUBinarnuDatoteku" obavlja sličnu funkcionalnost, samo što se upis vrši u binarnu datoteku, u koju je potrebno snimiti one podatke iz memorije koji su neophodni da bi se kasnije mogla pouzdano izvršiti rekonstrukcija stanja matrice iz pohranjenih podataka.Pri tome se podrazumijeva da je tip elemenata matrice neki skoro-POD tip podataka (u suprotnom, snimanje u binarnu datoteku najvnjerovatnije neće biti uspješno). U slučaju problema pri upisu, vrijedi isto kao kod prethodne metode. Dalje, metode "ObnoviIzTekstualneDatoteke" odnosno "ObnoviIzBinarneDatoteke" vrše obnavljanje sadržaja matrice na osnovu sačuvanog stanja u tekstualnoj odnosno binarnoj datoteci, pri čemu se prethodni sadržaj matrice uništava. U oba slučaja, ukoliko tražena datoteka ne postoji, treba baciti izuzetak tipa "logic_error" uz prateći tekst "Trazena datoteka ne postoji". Pri čitanju iz tekstualne datoteke, ukoliko datoteka sadrži podatke koji nisu u skladu sa tipom elemenata matrice, ukoliko podaci nisu razdvojeni zarezima, ili ukoliko različiti redovi imaju različit broj elementa, treba baciti isti izuzetak, uz prateći tekst "Datoteka sadrzi besmislene podatke" (s obzirom da u tekstualnoj datoteci nije pohranjena nikakva informacija o broju redova i kolona, datoteku ćete morati efektivno iščitati dva puta, prvi put da saznate broj redova i kolona, a drugi put da zaista pročitate vrijednosti elemenata, nakon što su obavljene odgovarajuće dinamičke alokacije). U slučaju bilo kakvih drugih problema pri čitanju, treba također baciti isti izuzetak, uz prateći tekst "Problemi pri citanju datoteke". Za slučaj čitanja iz binarne datoteke, u slučaju bilo kakvih problema (osim nepostojeće datoteke), treba baciti ovaj isti izuzetak.

Konačno, rečeno je da je u klasu "Matrica" potrebno dodati i novi konstruktor. Taj konstruktor će imati dva parametra, pri čemu je prvi parametar ime datoteke, a drugi je logička vrijednost koja

određuje da li će se konstrukcija objekta vršiti iz tekstualne datoteke (u slučaju kad taj parametar ima vrijednost "`false`") ili iz binarne datoteke (kada parametar ima vrijednost "`true`"). Ovaj konstruktor ponaša se identično kao što se ponašaju metode "`ObnoviIzTekstualneDatoteke`" odnosno "`ObnoviIzBinarneDatoteke`", samo se oslanja na činjenicu da se objekat tek stvara, tako da nema potrebe za oslobađanjem resursa koje je objekat prije toga koristio.

Obavezno napišite i kratki testni program u kojem ćete testirati novododane funkcionalnosti klase vezane za datoteke. Ostale funkcionalnosti klase ne morate testirati, s obzirom da se podrazumijeva da ste ih testirali ranije. Naravno, dozvoljena je upotreba i ostalih funkcionalnosti ukoliko su Vam potrebne za potrebe testiranje onoga što se traži da testirate (recimo, da formirate matrice koje želite snimiti, ili da ispišete sadržaj obnovljene matrice).

4. Potrebno je napraviti generičku funkciju "`SortirajBinarnuDatoteku`" koja omogućava sortiranje podataka pohranjenih u binarnoj datoteci *bez učitavanja sadržaja datoteke u memoriju*. Funkcija treba da ima sljedeći prototip:

```
template <typename TipElemenata>
  void SortirajBinarnuDatoteku(const char ime_datoteke[],
    std::function<bool(TipElemenata, TipElemenata)> kriterij
      = std::less<TipElemenata>());
```

Prvi parametar je ime datoteke, za koju se podrazumijeva da je organizirana kao kolekcija elemenata istog tipa "`TipElemenata`" (koji naravno mora biti neki skoro-POD tip), snimljenih u datoteku jedan za drugim. Pri tome je posve nebitno kako je ta kolekcija nastala (da li snimanjem elemenata jedan za drugim, "istresanjem" čitavog niza elemenata odjednom, ili na neki treći način). Drugi parametar je funkcija kriterija, koja radi na posve analogan način kao funkcija kriterija u bibliotečkoj funkciji "`sort`", pri čemu se može zadati bilo obična funkcija, bilo lambda funkcija, bilo funkcijski objekat (funktor), tačnije bilo šta što se može pozvati sa dva argumenta koji su tipa "`TipElemenata`" (ili se mogu konvertovati u taj tip) i koja vraća logičku vrijednost (ili nešto što se može interpretirati kao logička vrijednost). Recimo, ukoliko datoteka "`BROJEVI.DAT`" sadrži cijele brojeve, njen sadržaj možemo sortirati u opadajući poredak pozivom poput

```
SortirajBinarnuDatoteku<int>("BROJEVI.DAT", [](int x, int y) { return x > y; });
```

Nažalost, tip elemenata se mora eksplicitno specificirati (tj. nije moguće pisati samo nešto poput "`SortirajBinarnuDatoteku(...)`" umjesto "`SortirajBinarnuDatoteku<int>(...)`"), odnosno tip se ne može deducirati iz parametara lambda funkcije, jer je dedukcija moguća samo u slučaju potpunog slaganja tipa (drugi parametar je deklariran kao polimorfni funkcijski omotač, kojem se zaista može dodijeliti lambda funkcija, ali lambda funkcija nije polimorfni funkcijski omotač, pa nemamo potpuno slaganje po tipu). Koristeći funkcijske objekte iz bibiloteke "`functional`", prethodni poziv mogli smo kraće izvesti kao

```
SortirajBinarnuDatoteku<int>("BROJEVI.DAT", std::greater<int>());
```

Drugi parametar ima podrazumijevanu vrijednost "`std::less<TipElemenata>()`", tako da se može izostaviti ukoliko želimo sortiranje u podrazumijevani rastući poredak.

U slučaju da zadana datoteka ne postoji, treba baciti izuzetak tipa "`logic_error`" uz prateći tekst "Datoteka ne postoji". U slučaju bilo kakvih drugih problema pri čitanju ili pisanju datoteke treba baciti isti izuzetak uz prateći tekst "Problemi u pristupu datoteci".

Napisanu funkciju iskoristite u testnom programu koji će testirati rad ove funkcije. Taj testni program moraće prvo kreirati neku binarnu datoteku, zatim pozvati ovu funkciju, i onda konačno iščitati njen sadržaj, da se uvjerimo da je datoteka zaista sortirana. Najstrože je zabranjeno da funkcija "`SortirajBinarnuDatoteku`" učita sadržaj datoteke u memoriju, sortira ga u memoriji i zatim vrati sortirani sadržaj nazad. Sortiranje se mora obaviti isključivo manipuliranjem nad sadržajem datoteke. Kršenje ove zabrane rezultiraće *dodjelom nula poena na ovaj zadatak*. Za sortiranje koristite neki od naivnih algoritama sortiranja koji Vam je poznat. Vodite računa da, uz pogodne funkcije kriterija, ova funkcija mora biti u stanju da sortira obje datoteke "`BROJEVI.DAT`" i "`STUDENTI.DAT`" koje su demonstrirane na Predavanju 14_b!