Zadaci za Laboratorijsku vježbu 14 (opcionalna)

NAPOMENE:

- Ova vježba je opcionalna i radi se samostalno (za dodatne poene), bez pomoći i nadzora asistenata ili demonstratora. Boduje se samo ono što je urađeno do kraja dana u kojem je vježba postavljena.
- Studentima koji ne urade vježbu u planiranom vremenu, toplo se savjetuje da dovrše samostalno one zadatke koje nisu stigli uraditi za vrijeme vježbe, da bi se upoznali sa tehnikama rada sa datotekama (pogotovo ukoliko nemaju namjeru raditi posljednju zadaću).
- Tema vježbe je rad sa datotekama, stoga se studentima preporučuje da pogledaju kako se kreira datoteka na C9/Theia razvojnom okruženju i kako se smješta u trenutni direktorij (najlakše vam je ako smjestite datoteku u isti folder gdje je se nalazi i vaš "`main.cpp`" fajl). Datoteke možete kreirati na više načina: File > New, Alt + N, ili klikom na znak "+" poslije prikazanih tabova. Pazite da ispravno odaberete lokaciju datoteke. Najviše testova će testirati upravo rad sa datotekama!
- Imena datoteka na Unixoidnim sistemima (na kojima se, između ostalog, nalazi i C9/Theia, kao i autotester) su *case-sensitive* (dok npr. na Windows operativnim sistemima nisu), tj. pravi se razlika između velikih i malih slova! Obratite pažnju na to!
- Ne podrazumijeva se da datoteke na svome kraju sadrže novi red (mnogi studentski programi se oslanjaju na tu činjenicu prilikom čitanja podataka koji se završavaju novim redom)
- Ukoliko je moguće da vaš program prilikom izvršavanja modificira datoteku, napravite njenu kopiju da ne biste svaki put morali ponovo upisivati podatke, nego radite brisanje originalne datoteke i preimenujte kopiju (ili napravite još jednu kopiju) na originalno ime.

ZADACI:

1. Uz pomoć nekog tekstualnog editora kreirajte tekstualnu datoteku "`TEMPERATURE.TXT`" koja sadrži rezultate mjerenja temperature u nekoj meteorološkoj stanici. Datoteka je organizirana na sljedeći način. U prvom redu nalazi se datum mjerenja, u formatu *dan*/*mjesec*/*godina* (npr. 5/12/2014). U drugom redu nalazi se komentar, koji može biti proizvoljan tekst (to može biti npr. naziv mjernog mjesta, ili nešto drugo). U trećem redu nalaze se rezultati mjerenja temperature za taj dan (mjerenja se vrše nekoliko puta dnevno), koji su međusobno razdvojeni zarezima (iza posljednjeg rezultata *nema zareza*). Dalje se podaci ponavljaju za svaki od dana za koji su registrirani rezultati mjerenja. Slijedi primjer mogućeg izgleda ove datoteke:

   ```
   23/5/2016
   Meteoroloska stanica Bjelave
   7,12,16,20,18,13,6
   9/11/2014
   Mobilno mjerno mjesto Aerodrom, stanica II
   -1,3,6,10,8
   9/11/2014
   Meteoroloska opservatorija Bjelasnica
   -5,-2,0,3,1,2,-1,-4
   ```

   Zatim napravite program koji iščitava sadržaj ove tekstualne datoteke i kreira drugu tekstualnu datoteku "`IZVJESTAJ.TXT`" koja sadrži izvještaj o mjerenjima koji izgleda poput sljedećeg (prosječna temperatura se ispisuje fiksno na dvije decimale):

   ```
   Meteoroloska opservatorija Bjelasnica
   -------------------------------------
   Datum mjerenja: 9/11/2014
   Minimalna temperatura: -5
   Maksimalna temperatura: 3
   Prosjecna temperatura: -0.75

   Mobilno mjerno mjesto Aerodrom, stanica II
   ------------------------------------------
   Datum mjerenja: 9/11/2014
   Minimalna temperatura: -1
   Maksimalna temperatura: 10
   Prosjecna temperatura: 5.20
   ```

```
Meteoroloska stanica Bjelave
----------------------------
Datum mjerenja: 23/5/2016
Minimalna temperatura: 6
Maksimalna temperatura: 20
Prosjecna temperatura: 13.14
```

Spisak treba biti sortiran po datumu mjerenja, kao što je gore prikazano. U slučaju da su za dvije grupe mjerenja datumi isti, podaci za grupu sa manjom prosječnom temperaturom trebaju doći prije podataka za grupu sa većom prosječnom temperaturom. Pretpostavite da ulazna datoteka sadrži samo ispravne podatke, u ispravnom formatu (uključujući i ispravan format datuma). U slučaju da čitanje datoteke protekne korektno, program ne ispisuje ništa na ekran. S druge strane, ukoliko datoteka ne postoji, na ekran treba ispisati tekst "Datoteka TEMPERATURE.TXT ne postoji!" (sa prelaskom u novi red nakon ispisa), a ukoliko prilikom njenog čitanja dođe do bilo kakvih problema osim čitanja iza kraja datoteke (poput fizičkog oštećenja, nailaska na nenumeričke podatke prilikom čitanja numeričkih podataka, itd.) na ekran treba ispisati tekst "Problemi pri citanju datoteke TEMPERATURE.TXT" (također uz prelazak u novi red).

2. 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) i u buradima (za tečnosti). Sanduci i burad se modeliraju redom pomoću klasa "`Sanduk`" odnosno "`Bure`". Sanduk je opisan svojom težinom, nazivom predmeta koji se u njemu čuvaju (pretpostavlja se da jedan sanduk čuva samo istovrsne predmete), brojem predmeta koji se u njemu čuvaju i težinom predmeta koji se u njemu čuvaju (svi su iste težine). Bure je opisano svojom težinom, nazivom tečnosti koja se u njemu čuva, te težinom tečnosti koja se u njemu čuva. Informacijama o robi pohranjenoj u skladištu pristupa se pomoću dinamički alociranog niza pokazivača koji pokazuju na objekte tipa "`Sanduk`" ili tipa "`Bure`" (za tu svrhu, obje 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 (realan broj), naziv predmeta koji se čuvaju (tipa konstantni niz znakova), broj predmeta koji se u njemu čuvaju (cijeli broj) i težnina predmeta koji se u njemu čuvaju (realan broj), dok konstruktor klase "`Bure`" prima kao parametre težinu (realan broj), naziv tečnosti koja se čuva (konstantni niz znakova), te težinu tečnosti (realan broj). Pored konstruktora, obje klase "`Sanduk`" i "`Bure`" podržavaju metode "`DajTezinu`", "`DajUkupnuTezinu`" i "`Ispisi`" bez parmetara. Prva metoda daje težinu vlastitu težinu sanduka 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: Bure
Sadrzaj: Trofazni kataklingeri
Vlastita tezina: 10 kg
Ukupna tezina: 45 kg
```

Interfejs klase "`Skladiste`" treba sadržavati sljedeće elemente:

a) Konstruktor sa jednim parametrom koji predstavlja maksimalnu količinu objekata (sanduka odnosno buradi) koji se mogu pohraniti u skladištu. Ovaj konstruktor se ne smije koristiti za automatsku pretvorbu cijelih brojeva u objekte tipa "`Skladiste`".

b) Destruktor, koji oslobađa svu memoriju koji su objekti tipa "`Skladiste`" zauzeli tokom svog života.

c) Kopirajući konstruktor i preklopljeni kopirajući operator dodjele koji omogućavaju bezbjedno kopiranje i međusobno dodjeljivanje objekata tipa "`Skladiste`" zasnovano na strategiji dubokog kopiranja (nemojte zaboraviti da se radi o polimorfnoj kolekciji objekata, što zahtijeva izvjesnu podršku u ostalim klasama). Treba također predvidjeti i njihove optimizirane verzije za slučaj kopiranja privremenih objekata.

d) Metode "`DodajSanduk`" odnosno "`DodajBure`" koje kreiraju novi objekat tipa "`Sanduk`" odnosno "`Bure`" i pohranjuju ga u skladište. Parametri ovih metoda isti su kao i parametri konstruktora objekata tipa "`Sanduk`" odnosno "`Bure`". Ukoliko je skladište već popunjeno, treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Popunjeno skladiste".

e) Metode "`DajNajlaksi`" odnosno "`DajNajtezi`" koje vraćaju reference na najlakši odnosno najteži objekat (bure ili sanduk) 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".

f) Metodu "`BrojPreteskih`" koja 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. Metoda se mora moći pozvati i nad konstantnim objektom.

g) Metodu "`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.

h) 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`" ili "`B`" (za sanduk odnosno za bure) iza kojeg nakon jednog razmaka slijedi naziv predmeta ili tečnosti koje su pohranjene u sanduku odnosno buretu (npr. "`S Tepsije`" 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. "`1.2 50 0.35`") dok se za slučaj bureta nalazi težina bureta i težina tečnosti (npr. "`0.8 16.5`"). 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.

3. Proširite klasu "`Liga`" iz Zadatka 4 sa Laboratorijske vježbe 11. sljedećim elementima:

a) Metodom "`ObrisiSve`" koja briše kompletnu ligu, odnosno dovodi je u stanje kakvo je bilo nakon kreiranja objekta tipa "`Liga`".

b) Metodom "`SacuvajStanje`" koja sprema cijelo stanje lige u binarnu datoteku, pri čemu se kao parametar zadaje ime datoteke (tipa "`string`"). U slučaju bilo kakvih problema pri upisu, metoda baca izuzetak tipa "`logic_error`" uz prateći tekst "Problemi pri upisu u datoteku".

c) Metodom "`AzurirajIzDatoteke`" koja vrši ažuriranje stanja lige iz tekstualne datoteke čije se ime zadane kao parametar (tipa "`string`"). Datoteka je organizirana tako da prva dva reda sadrže imena timova koji su odigrali utakmicu, nakon čega u trećem redu slijedi rezultat utakmice u obliku poput "`3:2`" sa značenjem "3 postignuta gola za prvi tim, a 2 postignuta gola za drugi tim". Postojeći podaci u ligi se ne brišu, nego se samo stanje ažurira dodatnim informacijama. Ukoliko datoteka ne postoji, baca se izuzetak tipa "`logic_error`" uz prateći tekst "Datoteka ne postoji", a u slučaju bilo kakvih problema pri čitanju, metoda baca izuzetak istog tipa uz prateći tekst "Problemi pri citanju datoteke" (uključujući i slučaj kada datoteka sadrži nekonzistentne podatke, poput nenumeričkih podataka gdje se očekuje broj, ili negativan broj golova, itd.).

d) Konstruktorom koji obnavlja stanje lige iz binarne datoteke čije je ime zadano kao parametar konstruktora (tipa "`string`"). Za datoteku se očekuje da je onakva kakva je kreirana pod b). Ukoliko datoteka ne postoji, konstruktor baca izuzetak tipa "`logic_error`" uz prateći tekst "Datoteka ne postoji", a u slučaju bilo kakvih problema pri čitanju, metoda baca izuzetak istog tipa uz prateći tekst "Problemi pri citanju datoteke". Također, prije nego što se izvrši dinamička alokacija za podatke koji će biti obnovljeni, obavezno treba provjeriti da li je pročitana informacija o broju timova manja ili jednaka od kapaciteta lige. Ukoliko to nije slučaj (što se može desiti jedino ukoliko neko "podvali" neispravnu datoteku), treba baciti izuzetak tipa "`logic_error`" uz prateći tekst "Datoteka sadrzi fatalne greske". Ovo je neophodno uraditi, jer bi u suprotnom pokušaj obnove lige iz takve datoteke rezultirao krahom programa.

Ukoliko to smatrate potrebnim, dozvoljeno je izvršiti i neke dopune u klasi "`Tim`" (ali bez mijenjanja njenog interfejsa). Nakon izvršene modifikacije, izmijenite i glavni program tako da po završetku rada obavezno snima stanje lige u binarnu datoteku "`LIGA.DAT`". Na početku rada programa, ukoliko datoteka "`LIGA.DAT`" postoji, program treba da obnovi sadržaj lige iz ove datoteke, tako da program prosto nastavlja raditi sa istom ligom i istim rezultatima sa kojima je radio prilikom prethodnog pokretanja. U slučaju da datoteka "`LIGA.DAT`" ne postoji, program treba da kreira novu ligu, onakvu kakva je bila i u testnom programu Zadatka 4 sa Laboratorijske vježbe 11.

VAŽNA NAPOMENA: U ovom zadatku morate krenuti od svoje vlastite klase "`Liga`" kakvu ste imali na Laboratorijskoj vježbi 11, a ne od nečije tuđe (ovo će se *provjeravati*). Ukoliko niste ranije dovršili ovu klasu, morate je sada dovršiti, a ukoliko je niste ranije ni napisali, morate je sada napisati (a ne nipošto kopirati odnekud ili od nekoga). Ukoliko ne postupite ovako, zadatak će se tretirati kao prepisan.

4. Pretpostavimo da neka binarna datoteka sadrži snimljen sadržaj nekog niza realnih brojeva (tipa "`double`"). Napišite funkciju "`IzvrniDatoteku`" koja kao parametar prima ime datoteke (tipa "`string`"), a koja izvrće sadržaj datoteke "u ogledalu" tako da njen prvi element postane posljednji, drugi pretposljednji, itd. Pri tome, sve se mora izvoditi direktno nad sadržajem datoteke, odnosno nije dozvoljeno prethodno učitavanje sadržaja datoteke u memoriju. U slučaju da datoteka ne postoji, funkcija treba baciti izuzetak tipa "`logic_error`" uz prateći tekst "Datoteka ne postoji", a u slučaju bilo kakvih problema u čitanju, treba baciti isti izuzetak, uz prateći tekst "Problemi pri citanju datoteke". Obavezno napišite i testni programu kojem ćete testirati napisanu funkciju.