Zadaci za Tutorijal 11.

NAPOMENA: Studenti bi trebali da razmisle o zadacima koji će se raditi na tutorijalu prije nego što dođu na tutorijal, tako da već u startu imaju osnovne ideje kako riješiti zadatke. U suprotnom, rad na laboratorijskim vježbama neće biti produktivan. Prva dva zadatka traže svi zajedno svega nekoliko minuta za rješavanje (naravno, ko tačno zna šta treba da uradi). Posljednja dva zadatka su međutim dosta kompleksna, i mnogi studenti ih neće stići završiti za predviđeno vrijeme. Bez obzira na to, svi studenti bi trebali dovršiti samostalno sve ono što nisu stigli uraditi za vrijeme tutorijala, s obzirom da je razumijevanje ovog tipa zadataka od vitalne važnosti za polaganje drugog parcijalnog ispita.

1. Napravite klasu "NeobicnaKlasa" kod koje direktna i kopirajuća inicijalizacija cijelim brojem proizvode različite efekte. Konkretno, pokušaj direktne odnosno kopirajuće inicijalizacije treba da proizvedu na ekran ispis "Direknta inicijalizacija" odnosno "Kopirajuća inicijalizacija". Na primjer:

```
NeobicnaKlasa k1(5);          // Proizvodi ispis "Direktna inicijalizacija"
NeobicnaKlasa k2 = 5;         // Proizvodi ispis "Kopirajuca inicijalizacija"
```

2. Korisnik klase "StedniRacun" iz Zadatka 1. sa Tutorijala 10. želi da u svakom trenutku u programu može dobiti informaciju koliko je štednih računa (tj. objekata tipa "StedniRacun") ukupno kreirano od početka programa, i koliko ih je trenutno aktivno (tj. koliko ima trenutuno "živih" varijabli tipa "StedniRacun"). Tu informaciju želi da dobije putem dvije funkcije članice "DajBrojKreiranih" i "DajBrojAktivnih" koje se koriste kao u sljedećem primjeru:

```
StedniRacun s1, s2(100);
{ StedniRacun s3(50); }                          // Nakon } s3 više ne postoji...
std::cout << StedniRacun::DajBrojKreiranih()
  << " " << StedniRacun::DajBrojAktivnih();      // Ispisuje 3 2
```

   Proširite klasu "StedniRacun" razvijenu na prethodnom tutorijalu stvarima koje su neophodne da se podrži ova funkcionalnost.

3. Definirajte i implementirajte klasu "Tim" koja predstavlja jedan tim u fudbalskom prvenstvu. Klasa posjeduje privatne atribute "ime_tima", "broj_odigranih", "broj_pobjeda", "broj_nerijesenih", "broj_poraza", "broj_datih", "broj_primljenih" i "broj_poena". Ovi atributi redom predstavljaju ime tima (do 20 znakova), broj odigranih utakmica, broj pobjeda, broj neriješenih utakmica, broj poraza, ukupan broj datih i primljenih golova, te broj poena za razmatrani tim. Atribut "ime_tima" treba izvesti kao klasični niz znakova. Klasa treba da ima sljedeći interfejs:

```
Tim(const char ime[]);
void ObradiUtakmicu(int broj_datih, int broj_primljenih);
const char *DajImeTima() const;
int DajBrojPoena() const;
int DajGolRazliku() const;
void IspisiPodatke() const;
```

   Konstruktor treba da postavi ime tima na vrijednost zadanu parametrom, a sve ostale atribute klase na nulu. Ukoliko je ime tima predugačko, treba baciti izuzetak tipa "range_error" uz prateći tekst "Predugacko ime tima". Metoda "ObradiUtakmicu" treba da na osnovu rezultata utakmice koji joj se prenosi kao parametar (u vidu broja datih i primljenih golova sa posmatrane utakmice) ažurira ne samo atribute koje broje golove, nego i atribute koji broje odigrane utakmice, broj pobjeda, poraza i neriješenih utakmica, kao i broj bodova. Pri tome se za pobjedu dodjeljuju 3 boda, za neriješen rezultat 1 bod, dok se za poraz ne dodjeljuju nikakvi bodovi. U slučaju da se kao neki od parametara pošalje negativan broj, treba baciti izuzetak tipa tipa "range_error" uz prateći tekst "Neispravan broj golova". Metode "DajImeTima", "DajBrojPoena" i "DajGolRazliku" treba da vrate respektivno ime tima (tačnije, pokazivač na prvi znak imena), broj poena, kao i gol razliku (tj. razliku između ukupnog broja datih i primljenih golova) za posmatrani tim (ove metode treba implementirati unutar deklaracije klase). Konačno, metoda "IspisiPodatke" treba da ispiše na ekran sve podatke o timu u jednom redu, i to sljedećim redom: ime tima, broj utakmica, broj pobjeda, broj neriješenih utakmica, broj poraza, broj datih golova, broj primljenih golova i broj poena. Pri ispisu, za ime tima zauzmite prostor od 20 znakova, u kojem ime treba da bude ispisano *poravnato ulijevo*, a za sve brojne podatke prostor od 4 znaka na ekranu, pri čemu ispis svakog brojčanog podatka treba biti *poravnat udesno* unutar predviđenog prostora. Napišite i kratki testni program u kojem ćete demonstrirati napisanu klasu.

4. Napišite klasu "Liga" koja se oslanja na prethodno napisanu klasu "Tim". Klasa treba da ima privatne atribute "broj_timova" i "max_br_timova" koji čuvaju redom broj timova odnosno maksimalni dozvoljeni broj timova u ligi (atribut "max_br_timova" treba da bude konstantni atribut), kao i privatni atribut "timovi" koji će služiti za pristup dinamički alociranom nizu od "max_br_timova" elemenata, pri čemu je svaki element niza pokazivač na objekat tipa "Tim". Interfejs klase treba da izgleda ovako:

```
explicit Liga(int velicina_lige);
explicit Liga(std::initializer_list<Tim> lista_timova);
~Liga();
Liga(const Liga &l);
Liga(Liga &&l);
Liga &operator =(const Liga &l);
Liga &operator =(Liga &&l);
void DodajNoviTim(const char ime_tima[]);
void RegistrirajUtakmicu(const char tim1[], const char tim2[],
  int rezultat_1, int rezultat_2);
void IspisiTabelu();
```

Konstruktor treba da izvrši dinamičku alokaciju memorije za prihvatanje onoliko timova koliko je navedeno parametrom, dok destruktor treba da izvrši oslobađanje svih resursa koje je klasa "Liga" alocirala tokom svog rada. Sekvencijski konstruktor treba omogućiti konstrukciju objekata tipa "Liga" iz inicijalizacione liste čiji su elementi imena timova. Kopirajući konstruktor i kopirajući operator dodjele treba da se brinu za ispravno kopiranje i međusobno dodjeljivanje objekata tipa "Liga". Pri tome, kako je kapacitet lige (tj. maksimalni dozvoljeni broj timova) nepromjenljiv, treba podržati samo međusobnu dodjelu između dva objekta tipa "Liga" istog kapaciteta, u suprotnom treba baciti izuzetak tipa "logic_error" uz prateći tekst "Nesaglasni kapaciteti liga". Pomjerajući konstruktor i pomjerajući operator dodjele predviđeni su da djeluju kao optimizirane varijante kopirajućeg konstruktora i kopirajućeg operatora dodjele za slučaj kada se kopiraju privremeni objekata. Metoda "DodajNoviTim" kreira tim sa navedenim imenom (tj. dinamički kreira objekat tipa "Tim") i upisuje ga na prvo slobodno mjesto u ligu (tj. upisuje pokazivač na njega na odgovarajuće mjesto u nizu pokazivača na objekte tipa "Tim"). Pri tome se, naravno, broj timova u ligi povećava za jedinicu. U slučaju da je ime tima predugačko, baca se izuzetak (isti kakav baca izuzetak klase "Tim" u tom slučaju). U slučaju da je dostignut maksimalni broj timova, treba baciti izuzetak tipa "range_error" uz prateći tekst "Liga popunjena". Također, ukoliko se pokuša upisati tim čije ime već postoji, baca se izuzetak tipa "logic_error" uz prateći tekst "Tim vec postoji".

Ključna metoda klase je metoda "RegistrirajUtakmicu", čija prva dva parametra predstavljaju imena timova koji su odigrali utakmicu, dok su treći i četvrti parametar broj golova koji su dali prvi i drugi tim respektivno. Ova metoda treba da ažurira rezultate u tabeli za oba tima, odnosno da baci izuzetak tipa "logic_error" uz prateći tekst "Tim nije nadjen" ukoliko timovi sa navedenim imenima ne postoje u tabeli (tačnije, ukoliko makar jedan od timova nije nađen). Isto tako, u slučaju da se zada besmislen (negativan) broj golova, baca se isti izuzetak kao u metodi "ObradiUtakmicu" u klasi "Tim". Pri tome, ukoliko dođe do bacanja izuzetka, stanje lige mora biti identično kao da ova metoda uopće nije bila pozvana, tj. ne smije se desiti da dođe do parcijalne izmjene evidentiranih podataka (jaka sigurnost na izuzetke). Konačno, metoda "IspisiTabelu" treba da ispiše tabelu lige sortiranu u opadajućem poretku po broju bodova. Ukoliko dva tima imaju isti broj poena, tada prvo dolazi tim sa većom gol razlikom, a ako je i gol razlika ista, onda prije dolazi tim koji je prije po abecedi. Sortiranje vršite pozivom funkcije "sort", uz pogodno definiranu funkciju kriterija. Kako ova metoda nije planirana da bude inspektor, dozvoljeno je da njen poziv ostavi tabelu lige sortiranu, iako je ranije možda bila nesortirana. Ispis treba vršiti pozivom metode "IspisiPodatke" iz klase "Tim", tako da bi prikazana tabela trebala da ima izgled poput sljedećeg:

```
Čelik              18  11   5   1  34  10  38
Borac              18  10   3   5  27  13  33
Jedinstvo          18   9   4   5  33  20  31
Željeznicar        18   9   4   5  25  19  31
Zrinjski           18   8   6   4  23  24  30
Sarajevo           18   8   5   5  32  16  29
```

Napišite i naku testni program u kojem ćete demonstrirati razvijene klase "Tim" i "Liga". Testni program treba prvo kreirati malu fiksnu ligu sa 6 timova kao u tabeli prikazanoj iznad, nakon čega ulazi u petlju u kojoj se prikazuje trenutno stanje table, nakon čega se unose podaci o odigranoj utakmici, iza koje slijedi ponovo prikaz stanja tabele i tako u krug sve dok se ne zahtijeva kraj rada. Slijedi primjer dijaloga između korisnika i programa:

```
Borac                   0   0   0   0   0   0   0
Celik                   0   0   0   0   0   0   0
Jedinstvo               0   0   0   0   0   0   0
Sarajevo                0   0   0   0   0   0   0
Zeljeznicar             0   0   0   0   0   0   0
Zrinjski                0   0   0   0   0   0   0

Unesite ime prvog tima (ENTER za kraj): Celik
Unesite ime drugog tima: Zrinjski
Unesite broj postigutih golova za oba tima: 2 3

Zrinjski                1   1   0   0   3   2   3
Borac                   0   0   0   0   0   0   0
Jedinstvo               0   0   0   0   0   0   0
Sarajevo                0   0   0   0   0   0   0
Zeljeznicar             0   0   0   0   0   0   0
Celik                   1   0   0   1   2   3   0

Unesite ime prvog tima (ENTER za kraj): Jedinstvo
Unesite ime drugog tima: Cekrcici United
Unesite broj postigutih golova za oba tima: 3 2
Tim nije nadjen

Zrinjski                1   1   0   0   3   2   3
Borac                   0   0   0   0   0   0   0
Jedinstvo               0   0   0   0   0   0   0
Sarajevo                0   0   0   0   0   0   0
Zeljeznicar             0   0   0   0   0   0   0
Celik                   1   0   0   1   2   3   0

Unesite ime prvog tima (ENTER za kraj): Zeljeznicar
Unesite ime drugog tima: Celik
Unesite broj postigutih golova za oba tima: 1 1

Zrinjski                1   1   0   0   3   2   3
Zeljeznicar             1   0   1   0   1   1   1
Celik                   2   0   1   1   3   4   1
Borac                   0   0   0   0   0   0   0
Jedinstvo               0   0   0   0   0   0   0
Sarajevo                0   0   0   0   0   0   0

Unesite ime prvog tima (ENTER za kraj):
```

Prije nego što napišete testni program koji implementira dijalog popug opisanog, prvo istestirajte da li svi elementi napisane klase rade kako treba. Posebno testirajte da li rade dobro upravljački elementi klase poput destruktora, kopirajućeg konstruktora, kopirajućeg operatora dodjele, itd.

5. Izmijenite klasu "`Liga`" razvijenu u prethodnom zadatku, tako što će se za evidenciju pokazivača na dinamički alocirane objekte tipa "`Tim`" umjesto dinamički alociranog niza pokazivača koristiti vektor čiji su elementi pametni pokazivači na objekte tipa "`Tim`". Taj vektor će, naravno, biti privatni atribut klase (nazvaćemo ga isto "`timovi`", jer to traži najmanju prepravku klase). Privatni atributi "`broj_timova`" i "`max_br_timova`" više neće biti potrebni. Zaista, broj timova se može saznati testiranjem trenutne veličine vektora, dok maksimalan broj timova više nije potrebno zadavati, s obzirom da vektor može u toku rada po volji povećavati svoju veličinu (ograničeni smo isključivo količinom raspoložive memorije). Jedina izmjena u interfejsu klase biće u tome što konstruktoru više neće biti potreban parametar (zahvaljujući fleksibilnosti vektora, u ligu će se moći dodati onoliko timova koliko želimo, bez potrebe da unaprijed specificiramo njihov maksimalan broj).. Što se tiče semantičkih aspekata (tj. šta metode klase "`Liga`" treba da rade), jedina razlika je u tome što metoda "`DodajNoviTim`" ne mora provjeravati da li je dostignut maksimalan broj timova, s obzirom da ograničenje na maksimalan broj timova više ne postoji (osim ograničenja uvjetovanih količinom raspoložive memorije). Svi ostali aspekti funkcioniranja klase "`Liga`" ostaju neizmijenjeni. Obratite pažnju kakve izmjene traže upravljački elementi klase (destruktor, kopirajući konstruktor, itd.).