Zadaća 4.

Ova zadaća nosi ukupno 3,6 poena, pri čemu prvi i treći zadatak nose po 1,5 poen, a drugi zadatak nosi 0,6 poena. Sva tri zadatka se mogu uraditi na osnovu gradiva sa prvih 10 predavanja i pretpostavljenog predznanja iz predmeta "Osnove računarstva". Rok za predaju ove zadaće je četvrtak, 24. V 2012. (do kraja dana) i ne može se produžiti. Zadaće se predaju putem Zamgera.

1. Implementirajte jednostavan program koji olakšava vođenje administrativnih poslova u nekoj biblioteci. Program se zasniva na dvije klase "`Clan`" i "`Knjiga`". Primjerci ovih klasa modeliraju respektivno članove (klijente) biblioteke i knjige u biblioteci. Klasa "`Clan`" sadrži privatne atribute koji čuvaju informacije o evidencijskom (matičnom) broju člana, njegovom imenu i prezimenu, adresi, te broju telefona (svi ovi atributi su tipa "`string`", osim evidencijskog broja koji je cijeli broj). Interfejs klase sadrži konstruktor sa 5 parametara koji inicijalizira sve atribute na vrijednosti zadane parametrima, zatim trivijalne pristupne metode "`DajEvidencijskiBroj`", "`DajIme`", "`DajPrezime`", "`DajAdresu`" i "`DajTelefon`" koje prosto vraćaju vrijednosti odgovarajućih atributa, te metodu "`Ispisi`" koja ispisuje podatke o članu na ekran (format ispisa izaberite po želji). Klasa "`Knjiga`" sadrži privatne atribute koji čuvaju informacije o evidencijskom broju knjige, naslovu, imenu pisca, žanru i godini izdavanja, kao i informaciju o eventualnom zaduženju knjige. Evidencijski broj i godina izdavanja su cijeli brojevi, informacija o zaduženju čuva se kao pokazivač na člana koji je zadužio knjigu odnosno 0 ukoliko knjiga nije zadužena, dok su ostali atributi stringovnog tipa. Interfejs klase sadrži konstruktor sa 5 parametara koji inicijalizira sve atribute klase na vrijednosti zadane parametrima, osim informacije o zaduženju koja se postavlja tako da signalizira da knjiga nije zadužena. Pored konstruktora, interfejs klase sadrži trivijalne pristupne metode "`DajEvidencijskiBroj`", "`DajNaslov`", "`DajAutora`", "`DajZanr`", "`DajGodinuIzdanja`" i "`DajKodKogaJe`" koje vraćaju vrijednosti odgovarajućih atributa, te metode "`ZaduziKnjigu`", "`RazduziKnjigu`", "`DaLiJeZaduzena`" i "`Ispisi`". Metoda "`ZaduziKnjigu`" vrši zaduživanje knjige, a parametar joj je referenca na člana koji zadužuje knjigu. Metoda "`RazduziKnjigu`" nema parametara, a vrši razduživanje knjige. Metoda "`DaLiJeZaduzena`" također nema parametara i prosto vraća logičku vrijednost "tačno" ili "netačno" u ovisnosti da li je knjiga zadužena ili ne, dok metoda "`Ispisi`" vrši ispis podataka o knjizi na ekran (format ispisa izaberite po želji).

   Podaci o svakom članu odnosno svakoj knjizi čuvaju se u dinamički alociranim varijablama, kojima se pristupa pomoću dva vektora čiji su elementi pokazivači na članove (tj. na objekte tipa "`Clan`") odnosno pokazivači na knjige (tj. na objekte tipa "`Knjiga`"). Broj elemenata ovih vektora nije unaprijed određen, nego raste po potrebi sa dodavanjem novih članova odnosno knjiga u evidenciju. Program treba da korisniku ponudi izbor jedne od sljedećih opcija, koje se izvršavaju u petlji sve dok korisnik ne odluči da završi rad:

   a) Registriranje novog člana. Izborom ove opcije, program treba pitati korisnika o ličnim podacima člana, nakon čega se kreira odgovarajući objekat tipa "`Clan`" i upisuje u evidenciju. Program ne smije dozvoliti kreiranje člana čiji se evidencijski broj poklapa sa evidencijski brojem nekog od već postojećih članova.
   b) Pretraga članova. Izborom ove opcije, program treba pitati korisnika o evidencijskom broju, nakon čega pronalazi i ispisuje podatke o članu sa tim evidencijskim brojem, ili informaciju da takav član ne postoji.
   c) Izlistavanje članova. Izborom ove opcije, program treba da ispiše podatke o svim registriranim članovima, jedan za drugim.
   d) Registriranje nove knjige. Izborom ove opcije, program treba pitati korisnika o podacima o knjizi koja se registrira, nakon čega se kreira odgovarajući objekat tipa "`Knjiga`" i upisuje u evidenciju. Program ne smije dozvoliti kreiranje člana čiji se evidencijski broj poklapa sa evidencijski brojem nekog od već postojećih članova.

e) Zaduživnje knjige. Izborom ove opcije, program treba pitati korisnika o evidencijskom broju knjige, kao i o evidencijskom broju člana koji zadužuje knjigu, nakon čega se registrira da je knjiga zadužena kod navedenog člana. U slučaju da je neki od evidencijskih brojeva neispravan, ili ukoliko je knjiga već zadužena, treba prikazati odgovarajuću poruku o greški.
f) Razduživanje knjige. Izborom ove opcije, program treba pitati korisnika o evidencijskom broju knjige, nakon čega registrira da knjiga nije više zadužena. U slučaju da je evidencijski broj neispravan, ili ukoliko knjiga nije zadužena, treba prikazati odgovarajuću poruku o greški.
g) Pretraga knjiga. Izborom ove opcije, program treba pitati korisnika o evidencijskom broju, nakon čega pronalazi i ispisuje podatke o knjizi sa tim evidencijskim brojem, ili informaciju da takav član ne postoji. Ukoliko je knjiga zadužena, treba ispisati i osnovne podatke o članu kod koga se knjiga nalazi (evidencijski broj, ime i prezime). U suprotnom, treba ispisati da knjiga nije zadužena.
h) Izlistavanje knjiga. Izborom ove opcije, program treba da ispiše podatke o svim registriranim članovima, jedan za drugim.
i) Prikaz zaduženja. Izborom ove opcije, program treba pitati korisnika o evidencijskom broju člana, nakon čega ispisuje podatke o svim knjigama koje je zadužio navedeni član, ili informaciju da taj član nema zaduženja. U slučaju da je evidencijski broj neispravan, treba prikazati odgovarajuću poruku o greški.
j) Kraj programa. Izborom ove opcije, program završava sa radom.

2. Izmijenite program “`ucenici_obp.cpp`” priložen uz Predavanje 10 u skladu sa sljedećim zahtjevima:
   a) Atributi “`ime`” i “`prezime`” u klasi “`Ucenik`” sada će biti tipa “`string`” umjesto niza znakova. Kao posljedica toga, konstruktor klase “`Ucenik`” treba također da se promijeni, tim prije što više nije potrebno provjeravati da li je predjena maksimalna dozvoljena dužina imena odnosno prezimena.
   b) Ocjene se više neće čuvati u nizu cijelih brojeva, nego u vektoru cijelih brojeva (njegov kapacitet i dalje treba da bude “`BrojPredmeta`” elemenata).
   c) U klasi “`Razred`”, za evidenciju dinamički alociranih objekata tipa “`Ucenik`” umjesto dinamički alociranog niza pokazivača treba koristiti vektor čiji su elementi pokazivači na objekte tipa “`Ucenik`”. Pri tome, atributi koji čuvaju broj upisanih učenika i kapacitet razreda više neće biti potrebni. Zaista, broj upisanih učenika se može saznati testiranjem trenutne veličine vektora, dok kapacitet razreda više nije potrebno zadavati, s obzirom da vektor može u toku rada po volji povećavati svoju veličinu. Samim tim, konstruktoru više neće biti potreban parametar (zahvaljujući fleksibilnosti vektora, moći će se upisati onoliko učenika koliko želimo, bez potrebe da unaprijed specificiramo njihov maksimalan broj). Isto tako, metoda za evidentiranje novog učenika više ne treba provjeravati da li je dostignut maksimalan broj studenata, s obzirom da ograničenje na maksimalan broj učenika više ne postoji. Destruktor će i dalje biti potreban (da oslobodi sve dinamički alocirane objekte tipa “`Ucenik`” nakon što objekat tipa “`Razred`” prestane postojati).
   d) Potrebno je u klasu “`Razred`” dodati konstruktor kopije i preklopljeni operator dodjele, s ciljem da se omogući bezbjedno kopiranje i međusobno dodjeljivanje objekata tipa “`Razred`”, zasnovano na strategiji dubokog kopiranja. Naime, treba primijetiti da su dinamički alocirani objekti tipa “`Ucenik`” u vlasništvu klase “`Razred`”, iako nisu njen sastavni dio (razmislite dobro šta konstruktor kopije i preklopljeni operator dodjele tačno trebaju kopirati).

   Uvjerite se da uz ovakve modifikacije program i dalje radi ispravno. Posebno se trebate uvjeriti da konstruktor kopije i preklopljeni operator dodjele rade ispravno (za tu svrhu ćete morati ponešto dodati i u glavni program), i da nigdje ni pod kakvim okolnostima ne dolazi do curenja memorije.

3. Za potrebe rekonstrukcija Željeznica Bosne i Hercegovine, potrebno je napraviti program koji će vršiti automatsku najavu polazaka preko ozvučenja na željezničkoj stanici. Program treba najavljivati sve vozove koji odlase sa stanice u toku dana, kao i eventualna kašnjenja u polascima. Za tu svrhu, u programu je potrebno razviti dvije klase nazvane "`Polazak`" i "`RedVoznje`". Klasa "`Polazak`" vodi evidenciju o jednom polasku, dok klasa "`RedVoznje`" vodi evidenciju o svim polascima u toku dana. Klasa "`Polazak`" ima sljedeći interfejs:

```
Polazak(string odrediste, int broj_voza, int broj_perona,
  bool brzi_voz, int sat_polaska, int minute_polaska,
  int trajanje_voznje);
void PostaviKasnjenje(int kasnjenje);
bool DaLiKasni() const;
int DajTrajanjeVoznje() const;
void OcekivanoVrijemePolaska(int &sati, int &minute) const;
void OcekivanoVrijemeDolaska(int &sati, int &minute) const;
void Ispisi() const;
```

Objekti tipa "`Polazak`" čuvaju u sebi informaciju o nazivu odredišta, broju voza (cijeli broj sa maksimalno 5 cifara), broju perona (cijeli broj u opsegu od 1 do 6), informaciju o tome da li je voz brzi voz ili ne, vremenu polaska (sati i minute), trajanju vožnje u minutama, kao i informaciju o eventualnom kašnjenju (također u minutama). Konstruktor inicijalizira objekat u skladu sa vrijednostima zadanim parametrima konstruktora, osim informacije o eventualnom kašnjenju, koja se automatski inicijalizira na 0. Konstruktor treba da baci izuzetak ukoliko bilo koji od parametara ima besmislene vrijednosti. Metoda "`PostaviKasnjenje`" postavlja informaciju o eventualnom kašnjenju na vrijednost zadanu parametrom, dok metoda "`DaLiKasni`" omogućava da se sazna da li odgovarajuća vožnja kasni ili ne (metoda vraća logičku vrijednost "**`true`**" u slučaju kašnjenja, a logičku vrijednost "**`false`**" u suprotnom slučaju). Metoda "`DajTrajanje`" daje kao rezultat trajanje vožnje u minutama, dok metode "`OcekivanoVrijemePolaska`" i "`OcekivanoVrijemeDolaska`" omogućavaju da se sazna očekivano vrijeme polaska odnosno dolaska kada se uračuna iznos kašnjenja u odnosu na predviđeno vrijeme polaska/dolaska. Obje metode treba da smjeste sate i minute očekivanog vremena polaska/dolaska u dva cjelobrojna parametra koji joj se prosljeđuju. Konačno, metoda "`Ispisi`" treba da podrži ispis objekata tipa "`Polazak`" na ekran. U slučaju da se radi o polasku bez kašnjenja, ispis bi trebao da izgleda poput sljedećeg:

Lokalni voz broj 3423, odredište Zenica, polazi sa perona 2 u 15:40, a na odredište stiže u 17:10. Putnicima i voznom osoblju želimo ugodno putovanje.

U stvarnom sistemu, ovaj tekst bi se trebao proslijediti sklopu za sintezu govora koji bi emitirao govornu informaciju preko ozvučenja, ali za potrebe ovog zadatka zadovoljićemo se prostim ispisom na ekran.

U slučaju da se radi o polasku koji kasni, ispis bi trebao da izgleda poput sljedećeg:

Brzi voz broj 358, odrediste Ploče, sa predviđenim vremenom polaska u 6:40, kasni oko 35 minuta, te će poći oko 7.15. Očekuje se da voz stigne na odredište oko 10:30. Izvinjavamo se putnicima zbog eventualnih neugodnosti.

Klasa "`RedVoznje`" ima sljedeći interfejs:

```
explicit RedVoznje (int max_broj_polazaka);
~ RedVoznje ();
RedVoznje (const RedVoznje &red_voznje);
RedVoznje &operator =(const RedVoznje &red_voznje);
void RegistrirajPolazak(string odrediste, int broj_voza,
  bool brzi_voz, int broj_perona, int sat_polaska,
  int minute_polaska, int trajanje_voznje);
```

```
void RegistrirajPolazak(Polazak *polazak);
int DajBrojPolazaka() const;
int DajBrojPolazakaKojiKasne() const;
int DajProsjecnoTrajanjeVoznji() const;
Polazak &DajPrviPolazak() const;
Polazak &DajPosljednjiPolazak() const;
void IsprazniKolekciju();
void Ispisi(int sati, int minute) const;
```

Podaci o polascima se čuvaju u dinamički alociranim objektima tipa "`Polazak`", kojima se opet pristupa preko dinamički alociranog niza pokazivača na takve objekte. Alokacija tog niza pokazivača vrši se iz konstruktora. Parametar konstruktora predstavlja maksimalan broj polazaka koji se mogu registrirati. Destruktor oslobađa svu memoriju koja je zauzeta tokom života objekta, dok konstruktor kopije i preklopljeni operator dodjele omogućavaju bezbjedno kopiranje i međusobno dodjeljivanje objekata tipa "`RedVoznje`" korištenjem strategije dubokog kopiranja. Metoda "`RegistrirajPolazak`" podržana je u dvije verzije. Prva verzija kreira novi polazak u skladu sa parametrima (koji su identični kao kod konstruktora klase "`Polazak`") i registrira ga u kolekciji, dok druga verzija prosto kao parametar prihvata pokazivač na objekat tipa "`Polazak`" (za koji pretpostavljamo da je već na neki način kreiran) i registira ga u kolekciji. U oba slučaja, treba baciti izuzetak u slučaju da je dostignut maksimalan broj polazaka koji se mogu registrirati u kolekciji. Metode "`DajBrojPolazaka`", "`DajBrojPolazakaKojiKasne`" i "`DajProsjecnoTrajanjeVoznje`" daju respektivno ukupan broj registriranih polazaka, broj polazaka koji kasne, te prosječno trajanje svih registriranih vožnji u minutama. Metode "`DajPrviPolazak`" i "`DajPosljednjiPolazak`" daju kao rezultat prvi i posljednji polazak (tj. odgovarajući objekat tipa "`Polazak`") u toku dana (uključujući i eventualna kašnjenja). Obje metode vraćaju kao rezultat referencu da se izbjegne nepotrebno kopiranje objekata. Metoda "`IsprazniKolekciju`" uklanja sve registrirane polaske, tako da je nakon poziva ove metode kolekcija u identičnom stanju kakva je bila neposredno nakon kreiranja. Konačno, metoda "`Ispisi`" ispisuje informacije o svim polascima, počev od zadanog vremena do kraja dana, sortiranu po očekivanim vremenima polazaka (za sortiranje iskoristite funkciju "`sort`", uz pogodno definiranu funkciju kriterija koja treba biti privatna statička funkcija članica klase). Ispis pojedinačnih polazaka vrši se prostim pozivom metode "`Ispisi`" nad objektima tipa "`Polazak`" pohranjenim u kolekciji.

Sve metode implementirajte izvan deklaracije klase, osim trivijalnih metoda koje trebate implementirati direktno unutar deklaracije klase. Obavezno napišite i testni program u kojem ćete testirati sve elemente napisanih klasa. Posebno se trebate uvjeriti da konstruktor kopije i preklopljeni operator dodjele rade ispravno, kao i da ni u kom slučaju ne dolazi do curenja memorije.