Zadaća 4.

Ova zadaća nosi ukupno 4 poena, pri čemu prva dva zadatka nose po 1 poen, treći zadatak 1,5 poen i četvrti zadatak 0.5 poena. Svi zadaci se mogu uraditi na osnovu gradiva sa prvih jedanaest predavanja (prva dva zadatka ne traže poznavanje jedanaestog predavanja) i pretpostavljenog predznanja iz predmeta "Osnove računarstva". Rok za predaju ove zadaće je petak 23. V 2014. (do kraja dana) i ne može se produžiti. Zadaće se predaju putem Zamgera.

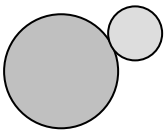
1. Definirajte i implementirajte klasu "`Krug`" koja omogućava čuvanje podataka koji opisuju jedan krug u ravni. Krug je opisan sa tri podatka: *x* i *y* koordinatom centra i poluprečnikom *r*, koji mora biti nenegativan broj (dozvoljava se da bude $r = 0$; u tom slučaju, krug se degenerira u tačku). Klasa treba da ima sljedeći interfejs:

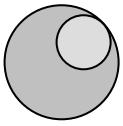
```
Krug(double r = 0);
Krug(double x, double y, double r = 0);
void PostaviPoziciju(double x, double y);
void PostaviPoluprecnik(double r);
double DajX() const;
double DajY() const;
double DajPoluprecnik() const;
double DajObim() const;
double DajPovrsinu() const;
void Ispisi() const;
void Transliraj(double delta_x, double delta_y);
void Rotiraj(double alpha);
void Rotiraj(double alpha, double x_c, double y_c);
friend bool DaLiSuIdenticni(const Krug &k1, const Krug &k2);
friend bool DaLiSuPodudarni(const Krug &k1, const Krug &k2);
friend bool DaLiSuKoncentrični(const Krug &k1, const Krug &k2);
friend bool DaLiSeDodirujuIzvana(const Krug &k1, const Krug &k2);
friend bool DaLiSeDodirujuIznutra(const Krug &k1, const Krug &k2);
friend bool DaLiSePreklapaju(const Krug &k1, const Krug &k2);
friend bool DaLiSeSijeku(const Krug &k1, const Krug &k2);
bool DaLiSadrzi(const Krug &k) const;
friend double RastojanjeCentara(const Krug &k1, const Krug &k2);
```

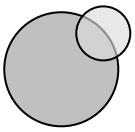
Konstruktor sa jednim parametrom kreira krug zadanog poluprečnika sa centrom u koordinatnom početku, dok konstruktor sa tri parametra kreira krug sa zadanom lokacijom centra i zadanim poluprečnikom. Oba konstruktora imaju podrazumijevanu vrijednost 0 za poluprečnik, tako da se mogu koristiti i bez parametara, odnosno sa dva parametra. Metode "`PostaviPoziciju`" i "`PostaviPoluprecnik`" omogućavaju naknadnu promjenu pozicije kruga (bez promjene poluprečnika) odnosno poluprečnika (bez promjene pozicije). Konstruktori kao i metoda "`PostaviPoluprecnik`" bacaju izuzetak tipa "`domain_error`" uz prateći tekst "Ilegalan poluprečnik" ukoliko se kao poluprečnik navede negativan broj. Metode "`DajX`", "`DajY`", "`DajPoluprecnik`", "`DajObim`" i "`DajPovrsinu`" vraćaju redom *x* koordinatu centra kruga, *y* koordinatu centra kruga, poluprečnik, obim i površinu kruga, dok metoda "`Ispisi`" ispisuje na ekran podatke o krugu u obliku "`{(x, y), r}`". Metoda "`Transliraj`" pomjera krug za iznos Δx u smjeru *x*-ose i iznos Δy u smjeru *y*-ose, pri čemu se vrijednosti Δx i Δy navode kao parametri. Metoda "`Rotiraj`" dolazi u dvije verzije. Prva verzija rotira krug za ugao α koji se zadaje kao parametar (u smjeru suprotnom od kazaljke na satu) oko koordinatnog početka, dok druga verzija omogućava da se zadaju i koordinave tačke (x_c, y_c) oko koje se vrši rotacija. Za one koji nisu dovoljno upućeni u analitičku geometriju, napomenimo da se rotacijom tačke (x, y) oko tačke (x_c, y_c) za ugao α dobija tačka (x', y') gdje su x' i y' dati kao $x' = x_c + (x - x_c)\cos\alpha - (y - y_c)\sin\alpha$ i $y' = y_c + (x - x_c)\sin\alpha + (y - y_c)\cos\alpha$. Ugao α se zadaje *u radijanima*. Naravno, prilikom translacije i rotacije, mijenja se samo pozicija kruga, dok njegov poluprečnik ostaje isti.

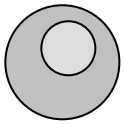
Predviđeno je i nekoliko prijateljskih funkcija koje ispituju međusobne odnose između krugova. Funkcija "`DaLiSuIdenticni`" vraća logičku vrijednost "tačno" ako i samo ako joj se kao parametri prenesu dva *identična* kruga (krugovi na *istoj poziciji* i sa *istim poluprečnikom*), inače

vraća logičku vrijednost "netačno". Funkcija "`DaLiSuPodudarni`" samo testira da li su krugovi *podudarni*, odnosno da li imaju *iste poluprečnike* (pri tome im se *pozicije mogu razlikovati*), dok funkcija "`DaLiSuKoncentricni`" testira da li krugovi imaju *zajednički centar*, dok im se *poluprečnici mogu razlikovati*. Dalje, slijede funkcije "`DaLiSeDodirujuIzvana`" odnosno "`DaLiSeDodirujuIznutra`" koje testiraju da li se krugovi *dodiruju*. Pretpostavlja se da se krugovi dodiruju ukoliko im *rubovi* (tj. kružnice kojima su omeđeni) imaju *tačno jednu zajedničku tačku*. Pri tome, dodir može biti *izvana*, kao na Slici 1, ili *iznutra*, kao na Slici 2. (dodir je izvana ukoliko je tim krugovima ta *zajednička tačka ujedno i jedina zajednička tačka*). Navedene funkcije upravo testiraju ta dva tipa dodira. Funkcija "`DaLiSePreklapaju`" testira da li *unutrašnjosti dva kruga* (unutrašnjost kruga čine sve njegove tačke koje *nisu na rubu*) *imaju zajedničkih tačaka*, dok funkcija "`DaLiSeSijeku`" testira da li se *rubovi dva kruga sijeku*, tj. imaju *tačno dvije zajedničke tačke*. Svaka dva kruga koja se sijeku također se i preklapaju, dok obrnuto ne mora vrijediti. Recimo, na Slici 3. imamo dva kruga koja se sijeku (i preklapaju), dok se krugovi na Slici 4. preklapaju, ali se ne sijeku. Konačno, predviđena je i funkcija članica (metoda) "`DaLiSadrzi`" koja testira da li se krug koji se zadaje kao njen parametar u potpunosti sadrži u krugu nad kojim je metoda pozvana (tj. da li je svaka njegova tačka ujedno i tačka kruga nad kojim je metoda pozvana), kao i prijateljska funkcija "`RastojanjeCentara`" koja vraća rastojanje između centara dva kruga koji joj se prenose kao parametri.


Slika 1


Slika 2


Slika 3


Slika 4

Napisanu klasu demonstrirajte u testnom programu koji traži da se tastature unese prirodan broj *n*, koji zatim treba dinamički alocirati niz od *n* krugova (tj. objekata tipa "`Krug`") koje treba postaviti na osnovu podataka koji se unose sa tastature (podaci o svakom krugu se unose posebno). Nakon okončanja unosa, program treba prvo translirati a zatim rotirati sve unesene krugove u skladu sa podacima koji se unose sa tastature. Za tu svrhu treba koristiti funkciju "`transform`" iz biblioteke "`algorithm`", pri čemu transformacionu funkciju koja se prosljeđuje funkciji "`transform`" treba izvesti kao lambda funkciju. Nakon obavljenih transformacija, treba ispisati podatke o svim unesenim krugovima nakon obavljenih transformacija. Podaci za svaki krug trebaju biti u posebnom redu. Ispis izvršite uz pomoć funkcije "`foreach`" i prikladne lambda funkcije. Potom treba ispisati podatke o krugu koji ima najveću površinu, za šta ćete iskoristiti funkciju "`max_element`" uz definiranje prikladne funkcije kriterija (ponovo kao lambda funkcije). Na kraju, program treba pronaći sve parove krugova koji se presjecaju i ispisati koji su to krugovi (ili ispisati obavijest da takvih krugova nema). To možete uraditi bez korištenja bibliotečkih funkcija, jer nema prikladnih funkcija za tu svrhu.

NAPOMENA 1: U testnom programu se očigledno ne testiraju sve metode klase. To ne znači da one ostale metode koje nisu predviđene u testnom programu ne moraju raditi ispravno!

NAPOMENA 2: Može se činiti da implementacija ove klase zahtijeva dobro poznavanje analitičke geometrije. Naprotiv, poznavanje formule za rastojanje dvije tačke i malo osnovne logike sasvim je dovoljno.

NAPOMENA 3: Mada ovaj zadatak ima dugačak opis, može se uraditi relativno brzo, jer sve tražene funkcije imaju vrlo kratke implementacije (jedan do dva reda koda).

2. Definirajte i implementirajte klasu "`Triangl`" koja omogućava čuvanje podataka koji opisuju jedan trougao/trokut (naziv "`Triangl`" za klasu izabran je radi uniformizacije imena klase koja je potrebna radi automatskog testiranja, a da se pri tome ne nameće jedan od naziva trougao ili trokut u slučaju da odabrani naziv nije u skladu sa Vašim jezičkim preferencijama). Pri tome se trougao (trokut) posmatra kao apstraktni geometrijski lik opisan isključivo dužinama svojih stranica, dok je njegov tačan položaj u ravni odnosno prostoru posve nebitan. Klasa treba da ima sljedeći interfejs:

```
explicit Triangl(double a);
Triangl(double a, double b);
Triangl(double a, double b, double c);
void Postavi(double a);
void Postavi(double a, double b);
void Postavi(double a, double b, double c);
static bool TestLegalnosti(double a, double b, double c);
double DajA() const;
double DajB() const;
double DajC() const;
double DajAlfa() const;
double DajBeta() const;
double DajGama() const;
double DajObim() const;
double DajPovrsinu() const;
void Skaliraj(double s);
void Ispisi() const;
friend bool DaLiSuPodudarni(const Triangl &t1, const Triangl &t2);
friend bool DaLiSuSlicni(const Triangl &t1, const Triangl &t2);
```

Konstruktor sa tri parametra kreira trougao čije su dužine stranica određene parametrima. Konstruktor sa dva parametra kreira pravougli trougao pri čemu parametri predstavljaju dužine kateta, dok konstruktor sa jednim parametrom kreira jednakostranični trougao pri čemu su dužine svih stranica jednake vrijednosti parametra. Metode "`Postavi`" (u tri varijante) načelno obavljaju isti zadatak kao i odgovarajući konstruktori, a omogućavaju naknadnu izmjenu podataka o trouglu. Svi konstruktori kao i metode "`Postavi`" trebaju baciti izuzetak tipa "`domain_error`" uz prateći tekst "Ilegalne stranice" ukoliko nije moguće kreirati trougao sa zadanim parametrima. Degenerirani slučajevi u kojima se trougao reducira na duž (recimo trougao sa stranicama dužine 1, 2 i 3, ili dužine 2, 2 i 0) ili na tačku (recimo, trougao sa dužinama stranica 0, 0 i 0) također *nisu dozvoljeni*. Statička metoda "`TestLegalnosti`" samo testira da li je moguće kreirati trougao sa stranicama koje su zadane kao parametri (bez bacanja izuzetaka) i vraća kao rezultat logičku vrijednost "tačno" ili "netačno" u zavisnosti da li je test uspio ili ne. Metode "`DajA`", "`DajB`" i "`DajC`" vraćaju kao rezultat dužine stranica trougla, metode "`DajAlfa`", "`DajBeta`" i "`DajGama`" vraćaju kao rezultat dužine odgovarajućih uglova/kutova trougla u *stepenima* (*alfa* je ugao naspram stranice *a*, itd.), dok metode "`DajObim`" i "`DajPovrsinu`" respektivno daju kao rezultat obim odnosno površinu trougla. Metoda "`Skaliraj`" skalira trougao sa faktorom koji je zadan kao parametar (tj. množi dužine svih stranica sa zadanom vrijednoću parametra). Faktor skaliranja mora biti pozitivan broj, inače treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Ilegalan faktor skaliranja". Metoda "`Ispisi`" ispisuje podatke o dužinama stranica, uglovima/kutovima, obimu i površini trougla (stil ispisa oblikujte po volji). Konačno, prijateljske funkcije "`DaLiSuPodudarni`" odnosno "`DaLiSuSlicni`" testiraju da li su dva trougla koja im se prenose kao parametri podudarna odnosno slična i vraćaju kao rezultat logičku vrijednost "tačno" ili "netačno" ovisno od rezultata testiranja. Dva trougla su podudarna ukoliko imaju identične stranice (koje ne moraju biti u istom redoslijedu tako da su, na primjer, podudarni trouglovi sa stranicama 5, 12 i 10 odnosno 10, 5 i 12), a slična ukoliko su im stranice proporcionalne (vrijedi ista primjedba).

Napisanu klasu demonstrirajte u testnom programu koji traži da se tastature unese prirodan broj *n*, koji zatim treba kreirati prazan vektor čiji su elementi pametni pokazivači na trouglove (tj. na objekte tipa "`Triangl`"). Nakon toga, sa tastature treba redom unositi podatke za *n* trouglova (podaci o svakom trouglu se unose posebno). Za svaki od trouglova, nakon obavljenog unosa treba dinamički kreirati odgovarajući trougao (tj. objekat tipa "`Triangl`") inicijaliziran u skladu sa unesenim podacima i pametni pokazivač na tako kreirani trougao ubaciti u vektor. Ukoliko korisnik zada vrijednosti stranica od kojih se ne može formirati trougao, treba ispisati poruku upozorenja i zatražiti novi unos podataka za isti trougao. Nakon okončanja unosa, program treba sortirati sve unesene trouglove u rastući poredak po obimu (tj. troguao sa manjim obimom dolazi prije trougla sa većim obimom) i ispisati podatke o svim trouglovima nakon obavljenog sortiranja. Za sortiranje obavezno koristiti bibliotečku funkciju "`sort`" uz pogodno definiranu funkciju kriterija kao lambda funkciju. Na kraju, program treba pronaći sve parove podudarnih i sličnih trouglova i ispisati koji su to trouglovi (ili obavijest da takvih parova nema).

3. Potrebno je napraviti program koji vrši najavu polazaka autobusa na displeju autobuske stanice. Program treba najavljivati sve polaske u toku dana, kao i eventualna kašnjenja u polascima. Za tu svrhu, u programu je potrebno razviti dvije klase nazvane "`Polazak`" i "`Polasci`". Klasa "`Polazak`" vodi evidenciju o jednom polasku, dok klasa "`Polasci`" vodi evidenciju o svim polascima u toku dana. Klasa "`Polazak`" ima sljedeći interfejs:

```
Polazak(std::string odrediste, std::string oznaka_voznje,
  int broj_perona, int sat_polaska, int minute_polaska,
  int trajanje_voznje);
void PostaviKasnjenje(int kasnjenje);
bool DaLiKasni() const;
int DajTrajanje() 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, oznaku vožnje (npr. "CTS 109"), broju perona (cijeli broj u opsegu od 1 do 15), 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 tipa "`domain_error`" uz odgovarajuće prateće tekstove (odredite ih po volji) ukoliko ma 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:

CTS 109 Bihać 7:30 15:10 5

Ovi podaci predstavljaju redom oznaku vožnje, naziv odredišta, vrijeme polaska, očekivano vrijeme dolaska i broj perona. Predvidite širinu od 10 mjesta za ispis oznake vožnje, 30 mjesta za naizv odredišta, po 10 mjesta za vrijeme polaska i dolaska, odnosno 8 mjesta za ispis perona. U slucaju da se radi o polasku koji kasni, ispis bi trebao da izgleda poput sljedećeg:

APM 314 Mostar 14:30 (Planirano 14:10, Kasni 20 min)

Oznaku vožnje, naziv odredišta i vrijeme polaska formatirajte kao i u prethodnom slučaju, a dopunske informacije pišite u produžetku iza vremena polaska.

Klasa "`Polasci`" ima sljedeći interfejs:

```
explicit Polasci(int max_broj_polazaka);
Polasci(std::initializer_list<Polazak> lista_polazaka);
~Polasci();
Polasci(const Polasci &polasci);
Polasci(Polasci &&polasci);
Polasci &operator =(const Polasci &polasci);
Polasci &operator =(Polasci &&polasci);
void RegistrirajPolazak(std::string odrediste,
  std::string oznaka_voznje, 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() 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. Predviđen je i sekvencijski konstruktor, koji omogućava kreiranje objekata tipa "`Polasci`" iz inicijalizacione liste čiji su elementi tipa "`Polazak`". Destruktor oslobađa svu memoriju koja je zauzeta tokom života objekta, dok kopirajući konstruktor i kopirajući operator dodjele omogućavaju bezbjedno kopiranje i međusobno dodjeljivanje objekata tipa "`Polasci`" korištenjem strategije dubokog kopiranja. Također su predviđeni i pomjerajući konstruktor odnosno operator dodjele koji optimiziraju postupak kopiranja u slučajevima kada se kopiraju privremeni objekti. 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 tipa "`range_error`" uz prateći tekst "Dostignut maksimalni broj polazaka" u slučaju da je dostignut maksimalan broj polazaka koji se mogu registrirati u kolekciji. Metode "`DajBrojPolazaka`", "`DajBrojPolazakaKojiKasne`" i "`DajProsjecnoTrajanjeVoznje`" daju redom ukupan broj registriranih polazaka, broj polazaka koji kasne, te prosječno trajanje svih registriranih vožnji u minutama. Metodu "`DajBrojPolazakaKojiKasne`" trebalo bi realizirati uz pomoć funkcije "`count_if`" iz biblioteke "`algorithm`" uz definiranje prikladne funkcije kriterija kao lambda funkcije. 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, i trebalo bi ih realizirati putem funkcija "`min_element`" i "`max_element`" iz biblioteke "`algorithm`", također uz odgovarajuće funkcije kriterija realizirane kao lambda funkcije. Metoda "`IsprazniKolekciju`" uklanja sve registrirane polaske, tako da nakon poziva ove metode kolekcija treba da bude u identičnom stanju kakva je bila neposredno nakon kreiranja. Konačno, metoda "`Ispisi`" ispisuje kompletan spisak svih polazaka, počev od zadanog vremena do kraja dana, sortiranu po očekivanim vremenima polazaka (za sortiranje iskoristite funkciju "`sort`", uz pogodno definiranu funkciju kriterija izvedenu kao lambda funkciju). U spisak treba dodati i prikladno zaglavlje, tako da bi ispis mogao izgledati poput sljedećeg:

Vožnja	***Odredište***	***Polazak***	***Dolazak***	***Peron***
CTS 109	***Bihać***	***7:30***	***15:10***	***5***
APM 314	***Mostar***	***14:30 (Planirano 14:10, Kasni 20 min)***		
SMT 291	***Čekrčići***	***15:35***	***16:20***	***12***

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 kopirajući i pomjerajući konstruktor kao i operatori dodjele rade ispravno, kao i da ni u kom slučaju ne dolazi do curenja memorije.

4. Izmijenite program iz prethodnog zadatka tako da se u klasi "`Polasci`" za evidenciju pokazivača na dinamički alocirane objekte tipa "`Polazak`" umjesto dinamički alociranog niza pokazivača koristiti vektor čiji su elementi pametni pokazivači na objekte tipa "`Polazak`", čime uklanjamo i ograničenje na maksimalno mogući broj polazaka koji se mogu registrirati, te rješavamo probleme vezane za oslobađanje memorije. Samim tim, konstruktor klase "`Polasci`" više neće imati parametar, dok metode za registraciju polazaka više ne trebaju provjeravati da li je dostignut

maksimalan broj polazaka, s obzirom da ograničenje na maksimalan broj polazaka više ne postoji. Također, druga verzija funkcije "`Polasci`" zahtijevaće kao parametar pametni a ne obični pokazivač na objekat tipa "`Polazak`". Razmislite sami šta treba da se desi sa destruktorom, kopirajućim i pomjerajućim konstruktorom i operatorima dodjele, te izvršite odgovarajuće izmjene. Obavezno testirajte da li sve radi ispravno nakon ovih izmjena.