Zadaća 4.

Ova zadaća nosi ukupno 4 poena. Prvi i treći zadatak nose po 1,3 poena, dok drugi i četvrti zadatak nose po 0,7 poena. Svi zadaci se mogu uraditi na osnovu gradiva sa prvih jedanaest predavanja i pretpostavljenog predznanja iz predmeta "Osnove računarstva". Rok za predaju ove zadaće je srijeda, 31. V 2017. do 10.00.

NAPOMENA: U slučaju da smatrate da Vam u zadacima trebaju neke pomoćne funkcije za realizaciju neophodnih funkcionalnosti (a trebale bi Vam, ako ne želite programe sa više od 500 linija koda), možete ih slobodno dodati u privatni dio klase. Međutim, interfejs klase *ne smijete mijenjati*, osim ukoliko se postavke zadatka jasno ne vidi da treba praviti izmjene u interfejsu klase.

1. Radi lakšeg definiranja tačka u ravni opisanih pomoću Dekartovih koordinata, na globalnom nivou uvedena je deklaracija

   ```
   typedef std::pair<double, double> Tacka;
   ```

   Definirajte i implementirajte klasu "Trougao" (ili "Trokut", ovisno od Vašeg jezičkog opredjeljenja) koja modelira trouglove (trokute) u ravni čija se tjemena nalaze u zadanim tačkama. Klasa treba da ima sljedeći interfejs:

   ```
   Trougao(const Tacka &t1, const Tacka &t2, const Tacka &t3);
   void Postavi(const Tacka &t1, const Tacka &t2, const Tacka &t3);
   void Postavi(int indeks, const Tacka &t);
   static int Orijentacija(const Tacka &t1, const Tacka &t2, const Tacka &t3);
   Tacka DajTjeme(int indeks) const;
   double DajStranicu(int indeks) const;
   double DajUgao(int indeks) const;
   Tacka DajCentar() const;
   double DajObim() const;
   double DajPovrsinu() const;
   bool DaLiJePozitivnoOrijentiran() const;
   bool DaLiJeUnutra(const Tacka &t) const;
   void Ispisi() const;
   void Transliraj(double delta_x, double delta_y);
   void Centriraj(const Tacka &t);
   void Rotiraj(const Tacka &t, double ugao);
   void Skaliraj(const Tacka &t, double faktor);
   void Rotiraj(double ugao);
   void Skaliraj(double faktor);
   friend bool DaLiSuIdenticni(const Trougao &t1, const Trougao &t2);
   friend bool DaLiSuPodudarni(const Trougao &t1, const Trougao &t2);
   friend bool DaLiSuSlicni(const Trougao &t1, const Trougao &t2);
   ```

   Konstruktor omogućava kreiranje trougla na osnovu koordinata tjemena koje se zadaju putem parametara. U slučaju da se sva tri tjemena nalaze na jednom pravcu, tako da se od njih ne može formirati trougao, treba baciti izuzetak tipa "domain_error" uz prateći tekst "Nekorektne pozicije tjemena" (za testiranje da li se tri tjemena nalaze na jednom pravcu ili ne, od koristi Vam može biti funkcija "Orijentacija" koju ćemo opisati kasnije). Za naknadnu izmjenu podataka o trouglu predviđene su dvije verzije metode Postavi". Prva prima iste parametre i obavlja isti zadatak kao i konstruktor, samo nad već postojećim objektom, dok druga omogućava da se zada koordinata samo jednog tjemena koje se mijenja, dok ostala dva tjemena ostaju nepromijenjena. Parametar "indeks" određuje koje se tjeme mijenja i može imati samo vrijednost 1, 2 ili 3, u suprotnom treba baciti izuzetak tipa "range_error" uz prateći tekst "Nekorektan indeks" (isto vrijedi za sve naredne funkcije koje također traže redni broj tjemena kao parametar).

   Statička funkcija članica "Orijentacija" testira kako su orijentirane tri tačke zadane kao parametri i vraća kao rezultat 1 ukoliko je orijentacija pozitivna (tj. ukoliko je kretanje od prve ka posljednjoj tački u smjeru suprotnom od smjera kazaljke na satu), -1 ukoliko je orijentacija negativna (kretanje u smjeru kazaljke na satu), te 0 ukoliko su tačke kolinearne tako da nema smisla govoriti o orijentaciji. Ukoliko su (x_1, y_1), (x_2, y_2) i (x_3, y_3) koordinate te tri tačke, test orjentacije se može obaviti testiranjem znaka izraza $x_1\,(y_2 - y_3) - x_2\,(y_1 - y_3) + x_3\,(y_1 - y_2)$ koji je pozitivan za pozitivnu orijentaciju, negativan za negativnu orijentaciju, te nula u slučaju da su razmatrane tačke kolinearne.

Predviđeno je više metoda pomoću kojih se mogu saznavati razne informacije o trouglu. Metoda "`DajTjeme`" vraća tjeme čiji je redni broj dat kao parametar, metoda "`DajStranicu`" vraća dužinu stranice koja se nalazi nasuprot tjemena čiji je redni broj dat kao parametar, dok metoda "`DajUgao`" (dopušteno je i "`DajKut`") daje iznos ugla trougla u radijanima koji se nalazi u tjemenu čiji je redni broj dat kao parametar (za računanje ugla najlakše je iskoristiti dužine stranica i kosinusnu teoremu, iako postoje i načini zasnovani na analitičkoj geometriji ili vektorskoj algebri direktno iz koordinata tjemena). Metoda "`DajCentar`" daje kao rezultat tačku koja predstavlja centar odnosno težište trougla (za trougao vrijedi da su koordinate njegovog centra aritmetičke sredine koordinata svih tjemena, ali to inače ne vrijedi generalno za druge poligone). Konačno, metode "`DajObim`" i "`DajPovrsinu`" daju respektivno obim odnosno površinu trougla. Površina se može izračunati direktno na osnovu koordinata kao jedna polovina apsolutne vrijednosti izraza $x_1 (y_2 - y_3) - x_2 (y_1 - y_3) + x_3 (y_1 - y_2)$. Dobro ste primijetili, ovo je isti izraz kao u funkciji za testiranje orijentacije(ubilo se za pomoćne funkcije).

Metoda "`DaLiJePozitivnoOrijentiran`" vraća logičku vrijednost "tačno" ukoliko je trougao nad kojim je pozvana pozitivno orijentiran (tj. ukoliko su mu tjemena posmatrana u redoslijedu zadavanja pozitivno orijentirana), a "netačno" u suprotnom. Metoda "`DaLiJeUnutra`" prima tačku kao parametar i vraća logičku vrijednost "tačno" ako i samo ako se razmatrana tačka nalazi unutar trougla (za tačku koja leži tačno na nekoj od stranica ne smatra se da leži unutar trougla). Mada problem određivanja da li je tačka unutar trougla ili ne izgleda jako težak, on u osnovi to nije. Naime, ukoliko su T_1, T_2 i T_3 koordinate tjemena trougla, a T tačka koja se testira, informacija o tome da li se tačka nalazi unutar trougla ili ne može se dobiti upoređujući orijentaciju trojki tačaka (T_1, T_2, T_3), (T_1, T_2, T), (T_2, T_3, T) i (T_3, T_1, T). Ovim ste dobili osnovnu ideju, a ostatak otkrijte sami (nacrtajte nekoliko crteža i lako ćete izvući zaključke). Podržana je i metoda "`Ispisi`" koja ispisuje informacije o tjemenima trougla u obliku "`((`x_1`,`y_1`),(`x_2`,`y_2`),(`x_3`,`y_3`))`".

Metoda "`Transliraj`" vrši tranlsaciju trougla za "`delta_x`" jedinica horizontalno i "`delta_y`" jedinica vertikalno. Predviđen je i specijalan slučaj translacije putem metode "`Centriraj`" koja obavlja takvu translaciju da se nakon nje centar (težište) trougla nađe u tački koja je zadana kao parametar. Metoda "`Rotiraj`" rotira trougao oko zadane tačke (centra rotacije) za zadani ugao. Za one koji ne znaju, a to su gotovo svi studenti koji će rješavati ovaj zadatak (što je velika sramota, ali za koju krivicu ne snose studenti), rotacijom tačke (x, y) oko tačke (x_c, y_c) za ugao α dobija se tačka $(x_c + (x - x_c)\cos\alpha - (y - y_c)\sin\alpha,\ y' = y_c + (x - x_c)\sin\alpha + (y - y_c)\cos\alpha)$. Metoda "`Skaliraj`" skalira trougao tako da mu se sve stranice produže sa zadanim faktorom, ali da trougao ostane i dalje u istom položaju, odnosno da stranice nakon skaliranja ostanu paralelne sa prethodnim stranicama. Prvi parametar je tzv. *centar skaliranja*, odnosno tačka čije pozicije nakon skaliranja ostaju nepromijenjene. Skaliranje se najlakše obavlja tako što se svaka tačka oblika (x, y) koja opisuje objekat zamjenjuje sa tačkom $(x_c + k\,(x - x_c),\ y_c + k\,(y - x_c))$ gdje je k faktor skaliranja. Dopušteno je da faktor skaliranja bude i negativan (u tom slučaju skalirani trougao biće zapravo još dodatno zarotiran za 180 °), ali ne i nula (takvo skaliranje bi reduciralo trougao na tačku). Stoga, ukoliko je faktor skaliranja jednak nuli, treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Nekorektan faktor skaliranja". Predviđene su i specijalni slučajevi metoda "`Rotiraj`" i "`Skaliraj`" sa samo jednim parametrom, pri čemu se tada za centar rotacije odnosno skaliranja uzima centar (težište) trougla.

Na kraju treba podržati još tri prijateljske funkcije (čija implementacija može biti vrlo jednostavna ako se ispravno odaberu uvjeti). Prijateljska funkcija "`DaLiSuIdenticni`" testira da li se trouglovi koji joj se prenose kao parametri poklapaju, tj. da li su im tjemena na istim mjestima u ravni, i vraća logičku vrijednost "**`true`**" ili "**`false`**", ovisno od rezultata testiranja. Vodite računa da to ne mora značiti da im tjemena sa istim rednim brojem moraju biti na istim mjestima, tako da je recimo trougao sa tjemenima u tačkama (1, 3), (5, 2) i (4, 4) tim redom identičan trouglovima sa tjemenima u tačkama (5, 2), (1, 3) i (4, 4) odnosno (4, 4), (5, 2) i (1, 3) tim redom. Pri tome, identični trouglovi ne moraju nužno imati istu orijentaciju (u navedenom primjeru, prvi trougao ima pozitivnu, a druga dva negativnu orijentaciju). Funkcija "`DaLiSuPodudarni`" testira da li su trouglovi koji se prenose kao parametri podudarni ili ne. Trouglovi su podudarni ukoliko se mogu dovesti na poklapanje translacijom i rotacijom. Na primjer, trougao sa tjemeinima u tačkama (2, 1), (6, 1) i (6, 4) podudaran je sa trouglom sa tjemenima u tačkama (1, 3), (1, 7) i (4, 3). Vodite računa da *nije dovoljno* da dva trougla imaju jednake dužine stranica da bi bili podudarni, nego da su neophodni još neki uvjeti (otkrijte ih sami). Na primjer, trougao sa tjemenima u tačkama (1, 3), (4, 3) i (4, 7) *nije poduraran* sa prethodna dva trougla (ne može se dovesti do poklapanja sa njima samo translacijom i rotacijom, nego je potrebno i *prevrtanje* – nacrtajte sliku), iako sva tri

trougla imaju dužine stranica 3, 4 i 5, ne nužno tim redom. Konačno, prijateljska funkcija "DaLiSuSlicni" ispituje da li su dva trouglaa slična ili ne. Dva trougla su slična ukoliko se mogu dovesti na poklapanje translacijom, rotacijom i skaliranjem. Uvjeti za sličnost su analogni uvjetima za podudarnost, samo što ovdje one stranice koje kod podudarnosti trebaju biti jednake ovdje ne moraju jednake, nego je dovoljno samo da budu međusobno proporcionalne (tj. odnos im treba da bude isti).

Napisanu klasu demonstrirajte u testnom programu koji traži da se tastature unese prirodan broj n, koji zatim treba kreirati prazan vektor od n pametnih pokazivača na objekte tipa "Trougao". Nakon toga, sa tastature treba redom unositi podatke za n trouglova (podaci o svakom trouglu se unose posebno, redom za prvo, drugo a zatim i za treće tjeme). Za svaki od trouglova, nakon obavljenog unosa treba dinamički kreirati odgovarajući trougao inicijaliziran u skladu sa unesenim podacima i pametni pokazivač na tako kreirani kreirani trougao ubaciti u vektor. Ukoliko korisnik zada tjemena od kojih se ne može formirati trougao, treba ispisati poruku upozorenja i zatražiti novi unos podataka za isti trougao. Nakon okončanja unosa, program treba prvo translirati sve trouglove u skladu sa podacima koji se unose sa tastature, a zatim ih rotirati oko njihovog centra te skalirati uzimajući prvo tjeme kao centar skaliranja, pri čemu se ugao rotacije i faktor skaliranja unose sa tastature. Za tu svrhu trebate koristiti funkciju "transform" iz biblioteke "algorithm", pri čemu transformacionu funkciju koja se prosljeđuje funkciji "transform" treba izvesti kao lambda funkciju. Nakon obavljenih transformacija, treba sortirati sve trouglove u rastući poredak po površini (tj. trougao sa manjom površinom dolazi prije trougla sa većom površinom), te ispisati podatke o svim trouglovima nakon obavljenog sortiranja. Podaci o svakom trouglu trebaju biti u posebnom redu. Za sortiranje obavezno koristiti bibliotečku funkciju "sort" uz pogodno definiranu funkciju kriterija kao lambda funkciju, a za ispis treba koristiti funkciju "foreach" i prikladnu lambda funkcije. Zatim treba ispisati podatke o trouglu koji ima najmanji obim, za šta ćete iskoristiti funkciju "max_element" uz definiranje prikladne funkcije kriterija (ponovo kao lambda funkcije). Konačno, na kraju, program treba pronaći sve parove identičnih, podudarnih i sličnih trouglova i ispisati koji su to trouglovi (ili obavijest da takvih parova nema). Ovo je okvirni opis šta testni program treba da radi, a precizan izgled dijaloga između korisnika i programa biće specificiran putem javnih autotestova.

2. Napravite klasu "GradjaninBiH" koja modelira jednog građanina Bosne i Hercegovine (misli se na građanina u pravnom smislu kao stanovnika države, a ne nužno osobu koja živi u gradu). Klasa treba da ima sljedeći interfejs

```
enum Pol {Musko, Zensko};
GradjaninBiH(std::string ime_i_prezime, long long int jmbg);
GradjaninBiH(std::string ime_i_prezime, int dan_rodjenja, int mjesec_rodjenja,
  int godina_rodjenja, int sifra_regije, Pol pol);
std::string DajImeIPrezime() const;
long long int DajJMBG() const;
int DajDanRodjenja() const;
int DajMjesecRodjenja() const;
int DajGodinuRodjenja() const;
int DajSifruRegije() const;
Pol DajPol() const;
void PromijeniImeIPrezime(std::string novo_ime);
```

Predviđena su dva konstruktora za konstrukciju objekata tipa "GradjaninBiH". U oba slučaja, prvi parametar je ime građanina (sa prezimenom). U prvoj varijanti, kao drugi parametar se zadaje jedinstveni matični broj građanina (JMBG). JMBG je 13-cifreni broj koji sadrži vrlo mnogo informacija o građaninu. Ako predstavimo JMBG u obliku $c_1c_2c_3c_4c_5c_6c_7c_8c_9c_{10}c_{11}c_{12}c_{13}$, tada c_1c_2 predstavlja dan rođenja, c_3c_4 mjesec rođenja, $c_5c_6c_7$ posljednje 3 cifre godine rođenja. c_8c_9 predstavljaju šifru regije rođenja (npr. 17 za Sarajevo), $c_{10}c_{11}c_{12}$ je kôd po kojem se međusobno razlikuju osobe rođene na isti dan u istoj regiji, pri čemu su za muške osobe rezervirani kodovi od 000 do 499, a za ženske osobe od 500 do 999, dok je c_{13} kontrolna cifra koja zavisi od ostalih 12 cifara prema formuli

$$c_{13} = 11 - (7\cdot(c_1+c_7) + 6\cdot(c_2+c_8) + 5\cdot(c_3+c_9) + 4\cdot(c_4+c_{10}) + 3\cdot(c_5+c_{11}) + 2\cdot(c_6+c_{12}))\,\%\,11$$

pri čemu se ukoliko se dobije $c_{13} = 11$, uzima se da je $c_{13} = 0$, a ako se dobije $c_{13} = 10$, smatra se da je kôd osobe neprikladan (pri dodjeljivanju JMBG tada se bira novi kôd). U slučaju da se zada JMBG koji nije validan, treba baciti izuzetak tipa "logic_error" uz prateći tekst "JMBG nije

validan". JMBG nije validan ukoliko prvih 7 cifara ne mogu formirati ispravan datum, ili ukoliko se posljednja cifra razlikuje od onoga što bi se dobilo gore prikazanom formulom, što uključuje i slučaj za koji bi formula dala da treba biti $c_{13} = 10$. Ukoliko se zadani JMBG poklapa sa JMBG nekog drugog građanina (tj. nekog drugog objekta tipa "`GradjaninBiH`"), treba baciti izuzetak tipa "`logic_error`" uz prateći tekst "Vec postoji gradjanin sa istim JMBG". Na primjer, ukoliko pokušamo izvršiti deklaracije

```
GradjaninBiH g1("Rambo Sulejmanovic", 1305956174235);
GradjaninBiH g2("Zan Klod Sejdic", 1305956174235);
```

druga deklaracija treba da baci izuzetak, zbog toga što već postoji građanin sa istim JMBG. Kopiranje i međusobno dodjeljivanje objekata ovog tipa treba zabraniti (s obzirom da bi kopija nekog objekta tipa "`GradjaninBiH`" svakako imala isti JMBG). Kako riješiti problem da li postoji neki drugi objekta tipa "`GradjaninBiH`" koji ima isti JMBG biće uskoro opisano.

Drugi konstruktor omogućava da se podaci o građaninu daju putem dana, mjeseca i godine rođenja, šifre regije rođenja, te pola. Ukoliko su podaci neispravni, treba baciti izuzetak tipa "`logic_error`" uz prateći tekst "Neispravni podaci". Podaci su ne.ispravni ukoliko dan, mjesec i godina rođenja nemaju smisla, ili ako šifra regije nije u opsegu od 0 do 99 (nisu ni sve šifre u ovom opsegu validne, ali to ćemo ovdje ignorirati). Ovaj konstruktor također na osnovu zadanih podataka treba da automatski kreira JMBG. Svi podaci za tu svrhu su poznati, osim kôda osobe. Za tu svrhu, treba uzeti prvi slobodni kôd, tj. najmanji kôd koji dosad nije iskorišten ni za jednu drugu osobu istog pola rođenu na isti dan u istoj regiji. Kako pronaći JMBG drugih građanina (tj. drugih već kreiranih objekata tipa "`GradjaninBiH`") biće također uskoro opisano.

Pored konstruktora, klasa "`GradjaninBiH`" posjeduje i pristupne funkcije "`DajImeIPrezime`", "`DajJMBG`", "`DajDanRodjenja`", "`DajMjesecRodjenja`", "`DajGodinuRodjenja`", "`DajSifruRegije`" i "`DajPol`" koje vraćaju informacije koje su jasne iz imena funkcija. Pri tome, treba imati na umu ukoliko je građanin zadan putem JMBG, nije moguće jednoznačno odrediti godinu rođenja, jer prva cifra godine rođenja nije sadržana u JMBG. Zbog toga, za realizaciju funkcije koja daje godinu rođenja uzeti pretpostavku da nijedan građanin nije stariji od 100 godina, kao i da je trenutno 2017. godina. Konačno, podržana je i funkcija "`PromijeniImeIPrezime`", koja omogućava promjenu imena građanina, a korisna je ukoliko recimo Brus Li Ramadanović poželi da promijeni ime i prezime u Čak Noris Bičakčić.

Očigledno, najveći problem u ovom zadatku je kako detektirati da li građanin kojeg kreiramo ima isti JMBG kao građani koji su već kreirani, odnosno kako konstruktor objekta tipa "`GradjaninBiH`" kojeg kreiramo može znati za druge objekte istog tipa. Jedno loše rješenje (koje nećete izvesti) je koristiti neki dijeljeni (statički) atribut koji bi bio recimo vektor pokazivača koji bi čuvao pokazivače na sve kreirane građane). Međutim, postoji rješenje koje je mnogo bolje sa aspekta utroška resursa (koje trebate izvesti u ovoj zadaći). Svaki objekat tipa "`GradjaninBiH`" će u sebi sadržavati jedan pokazivač na posljednjeg građanina koji je kreiran prije njega (osim prvog kreiranog građanina koji će na tom mjestu imati nul-pokazivač), tako da će svi kreirani objekti tipa "`GradjaninBiH`" faktički biti povezani u jednostruko povezanu listu (a sami objekti će biti čvorovi te liste). Pored toga, biće potreban i jedan dijeljeni statički atribut koji će sadržavati pokazivač na posljednjeg kreiranog građanina (ili nul-pokazivač ukoliko niti jedan građanin nije kreiran). Ovaj atribut je potreban da bismo znali gdje počinje "lanac" povezanih građana. Sad se test na jednakost JMBG izvodi tako što se prođe kroz čitavu listu i testira da li je JMBG građanina kojeg želimo kreirati jednak JMBG ma kojeg elementa liste. Ukoliko testiranje prođe uspješno, novokreirani građanin se također "uvezuje" u listu. Interesantno je da će klasa "`GradjaninBiH`" morati imati i destruktor, iako nigdje nema nikakve dinamičke alokacije memorije. Naime, kad objekat tog tipa prestane postojati, on mora sebe "isključiti" iz lanca. Obratite pažnju na specijalne slučajeve (tj. šta tačno treba ažurirati) kada se "isključuje" objekat koji se nalazi na jednom ili drugom kraju lanca. Na sličan način se izvodi i postupak pretrage koji su slobodni kôdovi u drugoj varijanti konstruktora koja treba automatski generirati JMBG.

Obavezno napišite i kratak testni program u kojem ćete testirati napisanu klasu (eventualni dijalozi između korisnika i programa biće specificirani javnim autotestovima). Za realizaciju programa najstrože je zabranjeno koristiti bibliotečke kontejnerske tipove podataka, kao što su vektori, dekovi, liste, skupovi, mape, itd. Nepoštovanje ove zabrane biće kažnjeno davanjem 0 poena na čitav zadatak!

3. Za potrebe evidencije prodaje stanova nekog stambenog fonda, potrebno je implementirati klase "Datum", "Kupac", "Stan", "Prodaja" i "Prodaje". Klasa "Datum" je minimalistička klasa sa sljedećim interfejsom:

```
Datum(int dan, int mjesec, int godina);
void Postavi(int dan, int mjesec, int mjesec);
int DajDan() const;
int DajMjesec() const;
int DajGodinu() const;
void Ispisi() const;
```

Klasa "Kupac" je također minimalistička, sa sljedećim interfejsom:

```
Kupac(const std::string &ime_i_prezime, const Datum &datum_rodjenja);
void Postavi(const std::string &ime_i_prezime, const Datum &datum_rodjenja);
std::string DajImePrezime() const;
Datum DajDatumRodjenja() const;
void Ispisi() const;
```

Konstruktor i metoda "Postavi" za obje ove klase omogućava inicijalizaciju i naknadnu izmjenu primjeraka ove klase, pri čemu se baca izuzetak tipa "domain_error" uz prateći tekst "Neispravan datum" odnosno "Neispravno ime i prezime" u slučaju da zadani podaci nisu legalni. Ime i prezime se smatraju nelegalnim ako sadrže bilo koji znak koji nije slovo, broj ili znak "crtica", razmak ili apostrof. Trivijalne pristupne metode "DajDan", "DajMjesec", "DajGodinu", "DajImePrezime" i "DajDatumRodjenja" samo vraćaju vrijednosti odgovarajućih atributa. Metoda "Ispisi" za klasu "Datum" ispisuje datum na ekran u obliku *dan/mjesec/godina* (npr. "23/5/2016"), dok istoimena metoda za osobu treba ispisati ime i prezime osobe i njen datum rođenja u obliku: *ime_i_prezime* (*dan/mjesec/godina*), npr. "Niko Nikic (17/5/1986)".

Klasa "Stan" ima sljedeći interfejs:

```
Stan(const std::string &adresa, int sprat, int broj_soba, bool namjesten,
  double kvadratura);
void Postavi(const std::string &adresa, int sprat, int broj_soba,
  bool namjesten, double kvadratura);
std::string DajAdresu() const;
int DajSprat() const;
int DajBrojSoba() const;
bool DajNamjesten() const;
double DajKvadraturu() const;
void Ispisi() const;
```

Konstruktor i funkcija "Postavi" postavljaju atribute klase na zadane vrijednosti. Ukoliko neki od parametara ima negativnu vrijednost, baca se izuzetak tipa "domain_error" uz prateći tekst "Neispravan unos podataka". Trivijalne pristupne metode samo vraćaju vrijednosti odgovarajućih atributa. Metoda "Ispisi" za klasu "Stan" ispisuje na ekran vrijednosti njenih atributa formatu "Stan se nalazi na adresi *adresa* na *sprat* spratu i ima *broj_soba* soba. Kvadratura stana je *kvadratura* i stan *je_ili_nije* namjesten". Na primjer:

```
Stan se nalazi na adresi Hamdije Cemerlica 14 na 5. spratu i ima 5 soba.
Kvadratura stana je 35.45 (m^2) i stan je namjesten.

Stan se nalazi na adresi Hamdije Cemerlica 14 na 1. spratu i ima 2 sobe.
Kvadratura stana je 39.21 (m^2) i stan nije namjesten.
```

Vodite računa o gramatičkoj korektnosti ispisa. Konkretno, treba voditi računa da li se broj sobe završava na 1, 2, 3 ili 4, tako da npr. za 3 sobe treba pisati "... i ima 3 sobe ..." a ne "... i ima 3 soba...".

Klasa "Prodaja" opisuje podatke o jednoj planiranoj prodaji, a ima sljedeći interfejs:

```
Prodaja(const std::string &ime_agenta_prodaje, double cijena_stana,
  const Datum &datum_prodaje, const Kupac &kupac_stana,
  const Stan &kupljeni_stan);
Prodaja(const std::string &ime_agenta_prodaje, double cijena_stana,
  int dan_prodaje, int mjesec_prodaje, int godina_prodaje,
  std::string &ime_kupca, const Datum &datum_rodjenja_kupca,
  const std::string &adresa_stana, int sprat_stana, int broj_soba,
  bool namjesten_stan, double broj_kvadrata);
void PromijeniKupca(const Kupac &novi_kupac);
```

```
void PromijeniStan(const Stan &novi_stan);
void PromijeniDatumKupovine(const Datum &novi_datum);
void PromijeniCijenuProdaje(const double &nova_cijena);
void PomjeriDanUnaprijed();
void PomjeriDanUnazad();
std::string DajImeAgenta() const;
std::string DajImeKupca() const;
Datum DajDatumProdaje() const;
double DajCijenuStana() const;
friend bool ProdatPrije(const Prodaja &p1, const Prodaja &p2);
friend bool SkupljiStan(const Prodaja &p1, const Prodaja &p2);
void Ispisi() const;
```

Konstruktori omogućavaju kreiranje jednog objekta tipa "`Prodaja`". U oba slučaja, ime agenta prodaje je prvi parametar, a cijena stana drugi. Podržana su dva konstruktora, jedan prima informacije o zakazanom datumu prodaje, kupcu stana i stanu o kojem je riječ putem objekata tipa "`Datum`", "`Kupac`" i "`Stan`" respektivno, dok drugi prima razbijene informacije o danu, mjesecu, godini prodaje, imenu i prezimenu kupca te njegovom datumu rođenja i naposlijetku o osobinama kupljenog stana.

Uz pomoć funkcija "`PromijeniKupca`", "`PromijeniDatumKupovine`", "`PromijeniCijenuKupovine`" i "`PromijeniStan`" mogu se naknadno promjeniti informacije o imenu kupca, odnosno zakazanom datumu prodaje, cijeni stana, ili stanu koji kupac namjerava kupiti. Pored toga, treba podržati i funkcije "`PomjeriDanUnaprijed`" i "`PomjeriDanUnazad`" koje pomjeraju zakazani datum prodaje za jedan dan unaprijed odnosno unazad.

Pristupne metode "`DajImeAgenta`", "`DajImeKupca`", "`DajDatumProdaje`" i "`DajCijenuStana`" vraćaju redom ime agenta prodaje, ime kupca stana, te datum i i cijenu stana koji se prodaje. Prijateljska funkcija "`ProdatPrije`" prima dva objekta tipa "`Prodaja`" kao parametre i vraća "**`true`**" ukoliko je prva prodaja zakazana prije druge, inače vraća "**`false`**". Konačno, funkcija "`Ispisi`" ispisuje podatke o prodaji na sljedeći način:

```
Ime agenta:                    Niko Nikic
Ime kupca:                     Mujo Mujic (21/9/1982)
Zakazani datum prodaje:        27/5/2017
Cijena stana:                  78985
Informacije o stanu:
Stan se nalazi na adresi Hamdije Cemerlica 14 na 5. spratu i ima 5 soba.
Kvadratura stana je 35.45 (m^2) i stan je namjesten.
```

Klasa "`Prodaje`" predstavlja kolekciju podataka o zakazanim prodajama, izvedenu kao dinamički alocirani niz pokazivača na objekte tipa "`Prodaja`". Ova klasa sadrži sljedeći interfejs:

```
explicit Prodaje(int max_broj_prodaja);
Prodaje(std::initializer_list<Prodaja> spisak_prodaja);
~Prodaje();
Prodaje(const Prodaje &prodaje);
Prodaje(Prodaje &&prodaje);
Prodaje &operator =(const Prodaje &prodaje);
Prodaje &operator =(Prodaje &&prodaje);
void RegistrirajProdaju(const std::string &ime_agenta_prodaje,
  double cijena_stana, const Datum &datum_prodaje, const Kupac &kupac_stana,
  const Stn &kupljeni_stan);
void RegistrirajProdaju(const std::string &ime_agenta_prodaje, int dan_prodaje,
  int mjesec_prodaje, int godina_prodaje, std::string &ime_kupca,
  const Datum &datum_rodjenja_kupca, const std::string &adresa_stana,
  int sprat_stana, int broj_soba, bool namjesten_stan, double broj_kvadrata);
void RegistrirajProdaju(Prodaja *prodaja);
int DajBrojProdaja() const;
int DajBrojProdajaNaDatum(const Datum &datum) const;
int DajBrojProdajaOdAgenta(const std::string &ime_agenta) const;
Prodaja &DajNajranijuProdaju();
Prodaja DajNajranijuProdaju() const;
Prodaja &DajNajskupljuProdaju();
Prodaja DajNajskupljuProdaju() const;
void IsprazniKolekciju();
void ObrisiNajranijuProdaju();
void ObrisiProdajeAgenta(const std::string &ime_agenta);
```

```
void ObrisiProdajeNaDatum(const Datum &datum);
void IspisiProdajeNaDatum(const Datum &datum) const;
void IspisiSveProdaje() const;
```

Konstruktor obavlja neophodnu alokaciju memorije, pri čemu parametar konstruktora predstavlja maksimalan broj prodaja koji se mogu registrirati. Predviđen je i sekvencijski konstruktor, koji omogućava kreiranje objekata tipa "`Prodaje`" iz inicijalizacione liste čiji su elementi tipa "`Prodaja`". 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 "`Prodaje`" korištenjem strategije dubokog kopiranja. Također su predviđeni i pomjerajući konstruktor odnosno pomjerajući operator dodjele koji optimiziraju postupak kopiranja u slučajevima kada se kopiraju privremeni objekti.

Metoda "`RegistrirajProdaju`" podržana je u tri verzije. Prve dvije verzije kreiraju novu prodaju u skladu sa parametrima (koji su identični kao kod konstruktora klase "`Prodaja`") i registriraju je u kolekciji, dok treća verzija prosto kao parametar prihvata pokazivač na objekat tipa "`Prodaja`" (za koji pretpostavljamo da je već na neki način kreiran) i registira ga u kolekciji. Registracijom se objekti tipa "`Prodaja`" predaju u vlasništvo objektu tipa "`Prodaje`", tako da je on odgovoran i za njihovo kasnije brisanje. U sva tri slučaja, treba baciti izuzetak tipa "range_error" uz prateći tekst "Dostignut maksimalni broj prodaja" u slučaju da je dostignut maksimalan broj prodaja koje se mogu registrirati.

Metode "`DajBrojProdaja`" i "`DajBrojProdajaNaDatum`" daju ukupan broj registriranih prodaja te broj prodaja zakazanih na zadani datum respektivno. Metodu "`DajBrojProdajaNaDatum`" trebalo bi realizirati uz pomoć funkcije "`count_if`" iz biblioteke "`algorithm`" uz definiranje prikladne funkcije kriterija kao lambda funkcije. Metoda "`DajNajranijuProdaju`" daje kao rezultat najranije zakazanu prodaju (tj. odgovarajući objekat tipa "`Prodaja`"). Treba podržati dvije verzije ove metode, pri čemu se za slučaj nekonstantnih objekata vraća referenca (da se izbjegne nepotrebno kopiranje i omogući izmjena vraćenog objekta). Isto vrijedi i za metodu "`DajNajskupljuProdaju`" koja kao rezultat daje najskuplju prodaju koja je zakazana (u odnosu na cijenu stana). Ove metode treba realizirati putem funkcije "`min_element`" i "`max_element`" iz biblioteke "`algorithm`", uz odgovarajuću funkciju kriterija realiziranu kao lambda funkcija. U slučaju da nema registriranih prodaja, obje funkcije trebaju baciti izuzetak tipa "`domain_error`" uz prateći tekst "Nema registriranih prodaja".

Metoda "`IsprazniKolekciju`" uklanja sve registrirane prodaje, tako da nakon poziva ove metode kolekcija treba biti u identičnom stanju kakva je bila neposredno nakon kreiranja. Metoda "`ObrisiNajranijuProdaju`" uklanja samo najraniju zakazanu prodaju (ova metoda se obično poziva nakon što izvrši zaključivanje kupovine). U slučaju da je kolekcija prazna, treba baciti izuzetak tipa "`range_error`" uz prateći tekst "Prazna kolekcija". Metoda "`ObrisiProdajeAgenta`" briše sve zakazane prodaje agenta čije je ime i prezime zadano kao parametar. Ukoliko nema registrirane nijedne prodaje sa zadanim imenom agenta, ne treba da se desi ništa. Metoda "`ObrisiProdajeNaDatum`" briše sve zakazane prodaje na datum koji joj je zadan kao parametar. Ukoliko nema registrirane nijedne prodajena dati datum, ne treba da se desi ništa.

Metoda "`IspisiProdajeNaDatum`" ispisuje informacije o svim prodajama zakazanim na zadani datum, sortirane u rastući poredak po imenu kupaca, dok metoda "`IspisiSveProdaje`" ispisuje informacije o svim zakazanim prodajama, u hronološkom redoslijedu (tj. sortirane po datumu u rastući poredak). U slučaju da ima više prodaja zakazanih na isti datum, takve prodaje između sebe trebaju biti sortirane po imenu kupaca. Pri tome, vodite računa da su ove metode realizirane kao inspektori, odnosno da ne smiju promijeniti sadržaj objekta nad kojim se pozivaju. Ispis pojedinačnih prodaja vrši se prostim pozivom metode "`Ispisi`" nad objektima tipa "`Prodaja`" pohranjenim u kolekciji.

Sve metode obavezno implementirajte izvan deklaracije klase, osim trivijalnih metoda koje možete implementirati direktno unutar deklaracije klase. Potrebno je napisati i testni program u kojem će korisnik putem menija birati koje transakcije želi da obavlja. Tačan izgled menija kao i dijaloga između korisnika i programa u raznim vrstama transakcija biće definirani putem jednog velikog javnog autotesta koji će biti postavljem vrlo brzo po izlasku ove zadaće. Međutim, u međuvremenu, svakako je potrebno prvo da sami istestirate funkcionalnost svih elemenata napisanih klasa. Posebno se trebate uvjeriti da kopirajući i pomjerajući konstruktor, te kopirajući i pomjerajući operator dodjele rade ispravno, kao da ni u kom slučaju ne dolazi do curenja memorije.

4. Izmijenite program iz prethodnog zadatka tako da se u klasi "`Prodaje`" za evidenciju pokazivača na dinamički alocirane objekte tipa "`Prodaja`" umjesto dinamički alociranog niza pokazivača koristiti vektor čiji su elementi pametni pokazivači na objekte tipa "`Prodaja`", čime uklanjamo i ograničenje na maksimalno mogući broj prodaja koje se mogu registrirati, te rješavamo probleme vezane za oslobađanje memorije. Samim tim, konstruktor klase "`Prodaje`" više neće imati parametar, dok metode za registraciju prodaja više ne trebaju provjeravati da li je dostignut maksimalan broj prodaja, s obzirom da ograničenje na maksimalan broj prodaja više ne postoji. Također, treća verzija funkcije "`RegistrirajProdaju`" zahtijevaće kao parametar pametni a ne obični pokazivač na objekat tipa "`Prodaja`". 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.