Zadaća 3.

Ova zadaća nosi ukupno 4 poena, pri čemu prvi zadatak nosi 0.7 poena, drugi 1 poen, treći i četvrti po 0.6 poena i peti 1.1 poen. Svi zadaci se mogu uraditi na osnovu gradiva sa prvih 9 predavanja i pretpostavljenog predznanja iz predmeta "Osnove računarstva". Rok za predaju ove zadaće je utorak, 8. V 2012. (do kraja dana) i ne može se produžiti. Zadaće se predaju putem Zamgera.

1. Dopunite program "`matrica_struct.cpp`" priložen uz Predavanje 7. sa dvije nove funkcije "`ProduktMatrica`" i "`StepenMatrica`". Funkcija "`ProduktMatrica`" prima dvije matrice kao parametre (matrice su definirane kao odgovarajuće strukture) i vraća njihov produkt kao rezultat, odnosno baca izuzetak ukoliko matrice nisu saglasne za množenje. Funkcija "`StepenMatrica`" prima matricu kao jedan parametar, kao drugi parametar prima prirodan broj *n*, i kao rezultat vraća matricu dignutu na *n*-ti stepen (u matričnom smislu) odnosno baca izuzetak ukoliko matrica nije kvadratna, s obzirom da je stepen matrice definiran samo za kvadratne matrice (pri realizaciji ove funkcije možete koristiti već napisanu funkciju "`ProduktMatrica`"). Također, proširite funkciju "`IspisiMatricu`" dodatnim parametrom "`treba_brisati`" tipa "**`bool`**". Ukoliko ovaj parametar ima vrijednost "**`true`**", funkcija treba da oslobodi prostor zauzet matricom koja joj je proslijeđena kao parametar, u suprotnom ne treba da radi ništa. Ovim se omogućava da možemo zadavati pozive poput

   ```
   IspisiMatricu(ZbirMatrica(a, b), 7, true);
   ```

   tako da se oslobađanje memorije koju je zauzela pomoćna matrica koja predstavlja zbir matrica može obaviti bez korištenja pomoćne promjenljive (kao u programu "`matrica_struct.cpp`"). Pri tome, definirajte da parametar "`treba_brisati`" ima podrazumijevanu vrijednost "**`false`**", tako da ga ne treba navoditi ukoliko nam brisanje ne treba. Napisane funkcije testirajte u glavnom programu na primjeru matrica čije dimenzije i elemente unosi korisnik putem tastature. U glavnom programu predvidite hvatanje svih izuzetaka koji bi eventualno mogli nastupiti. Također, dobro pazite da nigdje ne dođe do curenja memorije, ni pod kakvim okolnostima (u suštini, to je i *osnovni cilj zadatka*).

   Napomena: Da, ovaj zadatak je *već bio za zadaću u nekoliko prethodnih generacija*. Naravno, možete ga "pođoniti", ili Vam ga može uraditi neko drugi. Ali da znate, što bi rekao Balašević, *neko to od gore vidi sve*, i prije ili kasnije to će Vam se od glavu razbiti. Izuzetno je važno da barem ovaj zadatak *zaista uradite sami* (naravno, samostalno iste trebali uraditi i sve ostale zadatke, ali je izuzetno važno da baš ovaj zadatak ne prepišete ni .po koju cijenu). Ukoliko ne želite samostalno da uradite ovaj zadatak, *radije ga nemojte ni predavati*.

2. Zbog energetske krize, planiraju se restriktivna isključenja struje u pojedinim dijelovima grada. Grad je podijeljen u blokove, numerirane redom od 1 do N. Plan je da svaki M-ti blok počev od bloka 1 redom bude isključivan, nakon čega on ne dolazi u konkurenciju za ponovno isključivanje sve dok svaki blok ne bude barem jednom isključen. Pri tome se podrazumijeva da iza posljednjeg bloka ponovo dolazi blok sa rednim brojem 1, tako da se razbrajanje obavlja ciklično. Nakon što svi blokovi dođu na red, postupak se ponavlja ispočetka. Na primjer, ukoliko u gradu ima 10 blokova (sa rednim brojevima 1 – 10) i ukoliko je M = 4, redoslijed isključivanja je 1, 5, 9, 4, 10, 7, 6, 8, 3 i 2 (nacrtajte sliku), nakon čega ponovo započinje isti ciklus isključenja.

   Predložena strategija je očigledno pravična. Međutim, gospodin Hapo Kradibašić, ugledni predsjednik SPG-a (Stranke Pohlepnih Gulikoža), želi da kvart pod rednim brojem K u kojem se nalazi sjedište njegove stranke bude posljednji isključen, da ne bi došlo do remećenja priprema za predizbornu kampanju. Zbog toga on planira podmititi upravu Elektrodistribucije da odabere takav broj M da kvart pod rednim brojem K bude posljednji isključen.

   Vaš zadatak je da napravite program koji će gospodinu Kradibašiću pomoći da realizira svoju plemenitu zamisao. U programu treba implementirati dvije funkcije, "`Razbrajanje`" i

"OdabirKoraka". Funkcija "Razbrajanje" prima kao parametre broj blokova N i korak razbrajanja M, a kao rezultat daje vektor koji redom sadrži redne blokove blokova koji se isključuju poredane u redoslijedu isključivanja. Na primjer, ukoliko je N = 10 i M = 4, ova funkcija vraća vektor čiji su elementi redom 1, 5, 9, 4, 10, 7, 6, 8, 3 i 2. Pri tome, funkcija treba biti zasnovana na povezanim listama čvorova, koje se često primjenjuju upravo za rješavanje problema srodnih opisanom problemu. Konkretno, prvo ćete definirati čvornu strukturu "Blok" koja predstavlja jedan blok u gradu. Ona sadrži polje "redni_broj" tipa "**int**" i polje "sljedeci" koje je po tipu pokazivač na strukturu tipa "Blok". Polje "redni_broj" sadržavaće redni broj bloka, dok će polje "sljedeci" pokazivati na sljedeći blok. Funkcija "Razbrajanje" treba da kreira povezanu listu blokova (tj. čvorova tipa "Blok") čiji će redni brojevi biti postavljeni redom na vrijednosti od 1 do N, pri čemu će svaki blok pokazivati na blok sa narednim rednim brojem, osim posljednjeg bloka koji će pokazivati ponovo na prvi blok, čime se zapravo kreira krug blokoba (takve povezane liste u kojima posljednji čvor u listi pokazuje nazad na prvi čvor nazivaju se *kružne* ili *cirkularne liste*). Nakon što je formirana tražena lista, vrši se kretanje kroz listu, polazeći od prvog bloka, pri čemu se nakon svakih M napravljenih koraka odstranjuje onaj blok iz liste na kojoj se trenutno nalazimo i njegov redni broj smješta u izlazni vektor. Odstranjivanje se izvodi tako što se pokazivač "sljedeci" bloka koji prethodi bloku na kojem se trenutno nalazimo preusmjerava tako da ne pokazuje više na blok na kojem se trenutno nalazimo nego na blok koji slijedi iza njega (čime se blok efektivno odstranjuje iz razmatranja) i prelazimo na sljedeći blok. Tom prilikom, pomoću operatora "**delete**" potrebno je izbrisati čvor koji odgovara odstranjenom bloku, da ne zauzima više memoriju. Postupak se ponavlja dok se ne eliminira svih N blokova, nakon čega vraćamo vektor u kojem smo zapamtili redoslijed odstranjivanja blokova.

Već je rečeno da program pored funkcije "Razbrajanje", treba implementirati i funkciju "OdabirKoraka". Ova funkcija kao parametre prima broj blokova N i redni broj odabranog bloka K, a rezultat daje korak razbrajanja M koji treba uzeti da bi posljednji isključeni blok bio upravo blok sa rednim brojem K. Ukoliko takvih koraka ima više, funkcija vraća najmanji mogući korak, a ukoliko slučajno takva vrijednost koraka ne postoji, funkcija kao rezultat vraća 0 (postavljaču zadatka nije poznato može li se ovo ikada desiti, ali predvidimo i ovo za svaki slučaj). Rad ove funkcije zasniva se na pozivanju funkcije "Razbrajanje" za razne vrijednosti M sve dok se ne pronađe potrebni korak. Napisane funkcije demonstrirajte u testnom programu koji za zadane vrijednosti N i K ispisuje traženi korak razbrajanja koji gospodin Hapo Kradibašić treba saopćiti Elektrodistribuciji da bi postigao željeni cilj,

Napomena: Za izradu programa *nije dozvoljeno* koristiti koncepte koji nisu obrađivani na predavanjima niti na kursu OR (recimo, tipove "list", "set" ili "map", razne bibliotečke funkcije koje nisu obrađivane, itd.).

3. Za potrebe neke aplikacije za dvodimenzionalnu kompjutersku grafiku, potrebno je razviti klasu "Tacka" koja predstavlja tačku u ravni, sa sljedećim interfejsom:

```
Tacka();
Tacka(double x, double y);
void Postavi(double x, double y);
void PostaviPolarno(double ro, double theta);
double DajX() const;
double DajY() const;
double DajRo() const;
double DajTheta() const;
void PostaviX(double x);
void PostaviY(double y);
bool DaLiJeKoordinatniPocetak() const;
void Transliraj(double delta_x, double delta_y);
void Rotiraj(double alpha, const Tacka &centar = Tacka());
friend bool DaLiSuIdenticne(const Tacka &t1, const Tacka &t2);
friend double Rastojanje(const Tacka &t1, const Tacka &t2);
```

Konstruktor bez parametara kreira tačku u koordinatnom početku, dok konstruktor sa dva parametra kreira tačku na osnovu zadanih Dekartovih koordinata x i y. Istu stvar radi i metoda "`Postavi`" (osim što omogućava naknadnu promjenu pozicije tačke. Metoda "`PostaviPolarno`" radi sličnu stvar samo na osnovu polarnih koordinata ρ i θ (podsjetimo se da je veza između Deka rtovih i polarnih koordinata data formulama $x = \rho\cos\theta$, $y = \rho\sin\theta$ odnosno $\rho = \sqrt{x^2 + y^2}$, $\theta = \text{atan2}(x, y)$, gdje je "atan2" funkcija srodna funkciji "arkus tangens", a podržana je u jezicima C/C++ baš pod tim imenom). Metode "`DajX`", "`DajY`", "`DajRo`" i "`DajTheta`" vraćaju kao rezultat odgovarajuće Dekartove odnosno polarne koordinate tačke, dok metode "`PostaviX`" i "`PostaviY`" omogućavaju promjenu x odnosno y koordinate tačke (bez promjene preostale koordinate). Metoda "`DaLiJeKoordinatniPocetak`" vraća logičku vrijednost "tačno" ili "netačno" u ovisnosti da li tačka na koju je metoda primijenjena predstavlja koordinatni početak ili ne. Metoda "`Transliraj`" pomjera tačku za iznos Δx u smjeru x-ose i iznos Δy u smjeru y-ose, pri čemu se vrijednosti Δx i Δy navode kao parametri, dok metoda "`Rotiraj`" rotira tačku za ugao α koji se zadaje kao prvi parametar (u smjeru suprotnom od kazaljke na satu) oko tačke koja se zadaje kao drugi parametar. Drugi parametar kao podrazumijevanu vrijednost ima tačku kreiranu u koordinatnom početku, čime je postignuto da se može zadati samo jedan parametar (pri čemu se tada rotacija vrši oko koordinatnog početka). Za one koji nisu dovoljno upućeni, recimo da se rotacijom tačke (x, y) oko tačke (x_c, y_c) za ugao α dobija tačka (x', y') gdje je $x' = x_c + (x - x_c)\cos\alpha - (y - y_c)\sin\alpha$, $y' = y_c + (x - x_c)\sin\alpha + (y - y_c)\cos\alpha$. Konačno, klasa podržava i dvije prijateljske funkcije. Funkcija "`DaLiSuIdenticne`" vraća logičku vrijednost "tačno" ili "netačno" u ovisnosti da li su tačke koje su joj proslijeđene kao parametri identične ili ne, dok funkcija "`Rastojanje`" vraća kao rezultat rastojanje između tačaka koje su joj proslijeđene kao parametri.

Implementirajte klasu sa navedenim osobinama, pri čemu ćete uzeti da su atributi klase Dekartove koordinate tačke x i y. Napisanu klasu demonstrirajte u testnom programu koji traži da se tastature unese prirodan broj n, koji zatim treba dinamički alocirati niz od n tačaka (tj. objekata tipa "`Tacka`") koje treba inicijalizirati na osnovu podataka koji se unose sa tastature (podaci o svakoj tački se unose posebno). Nakon okončanja unosa, program treba prvo translirati a zatim rotirati sve unesene tačke u skladu sa podacima koji se unose sa tastature, te ispisati vrijednosti x, y, ρ i θ za sve unesene tačke nakon obavljenih transformacija. Na kraju, program treba ispitati da li među unesenim tačkama ima jednakih tačaka (rezultat ispitivanja treba prikazati na ekranu), te pronaći par tačaka koje se nalaze na najmanjem međusobnom rastojanju i ispisati koje su to tačke.

NAPOMENA: 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.

4. Pri razvoju neke hipotetičke aplikacije koja koristi klasu "`Tacka`" razvijenu u prethodnom zadatku, uočeno je da će neki vitalni algoritmi u aplikaciji mnogo brže raditi ukoliko se koordinate tačke interno čuvaju u polarnom koordinatnom sistemu, te da je pogodnije kao atribute klase čuvati njene polarne koordinate ρ i θ. Implementirajte ponovo klasu "`Tacka`" iz prethodnog zadatka koristeći ovako izmijenjen dizajn, ali tako da nova klasa zadrži isti interfejs, odnosno da korisnici klase (bar sa aspekta njene upotrebe) ne primijete da je njena implementacija izmijenjena. Posebno, testni program koji je razvijen u prethodnom zadatku treba da radi ispravno bez ikakvih izmjena sa novonapisanom verzijom klase "`Tacka`".

   NAPOMENA: Ukoliko prilikom rješavanja Zadatka 3. pametno napišete pojedine metode klase, nećete ih uopće morati mijenjati u novoj verziji klase (ideja je pisati metode tako da se što više oslanjaju na druge napisane metode). U suprotnom, možda ćete morati iznova napisati sve metode, što nije baš relaksirajuće.

5. Definirajte i implementirajte klasu "`Trougao`" koja omogućava čuvanje podataka koji opisuju jedan trougao/trokut. Pri tome se trougao 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:

```
Trougao(double a);
Trougao(double a, double b);
Trougao(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);
void OcitajStranice(double &a, double &b, double &c) const;
void OcitajUglove(double &alfa, double &beta, double &gama) const;
double DajObim() const;
double DajPovrsinu() const;
void Ispisi() const;
friend bool DaLiSuPodudarni(const Trougao &t1, const Trougao &t2);
friend bool DaLiSuSlicni(const Trougao &t1, const Trougao &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 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. Metoda "`OcitajStranice`" smješta vrijednosti stranica u promjenljive koje su proslijeđene kao parametri, dok funkcija "`OcitajUglove`" smješta vrijednosti odgovarajućih uglova/kutova izražene u radijanima u promjenljive koje su proslijeđene kao parametri (pri tome je *alfa* ugao naspram stranice *a*, itd.). Metode "`DajObim`" i "`DajPovrsinu`" respektivno daju kao rezultat obim odnosno površinu trougla. Metoda "`Ispisi`" ispisuje podatke o dužinama stranica, uglovima, 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 vektor koji sadrži *n* pokazivača na trouglove (tj. objekte tipa "`Trougao`"). Nakon toga, sa tastature treba unijeti podatke za *n* trouglova (podaci o svakom trouglu se unose posebno), dinamički kreirati odgovarajuće trouglove (tj. objekte tipa "`Trougao`") inicijalizirane u skladu sa unijetim podacima i povezati ih sa odgovarajućim pokazivačima unutar vektora. 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 opadajući poredak po površini (tj. troguao sa većom površinom dolazi prije trougla sa manjom površinom) i ispisati podatke o svim trouglovima nakon obavljenog sortiranja. Za sortiranje obavezno koristiti bibliotečku funkciju "`sort`" uz pogodno definiranu funkciju kriterija. Na kraju, program treba pronaći sve parove podudarnih i sličnih trouglova i ispisati koji su to trouglovi (ili obavijest da takvih parova nema).