# Zadaća 2.

***Ova zadaća nosi ukupno 4,5 poena, od kojih prvi zadatak nosi 1,3 poena, a ostali po 0,8 poena. Svi zadaci se mogu uraditi na osnovu gradiva s prvih 6 predavanja i pretpostavljenog predznanja iz predmeta "Osnove računarstva". Svi zadaci osim prvog mogu se uraditi tako da budu prilično kratki, a i prvi zadatak se može uraditi u manje od 150 linija kôda ukoliko se malo inteligentnije osmisle neke stvari, a ne samo nabacuje nepregledna sekvenca if-naredbi (što ne vodi ničemu dobrom). Rok za predaju ove zadaće je utorak, 16. IV 2024. do kraja dana.***

**VEOMA VAŽNO**: U svim zadacima, za indeksaciju vektora, dekova, modernih nizova i stringova, koristite funkciju "`at`" umjesto operatora "`[]`" (tj. umjesto "`a[i]`" odnosno "`a[i][j]`" pišite "`a.at(i)`" odnosno "`a.at(i).at(j)`". To je jedini način da se sa sigurnošću utvrdi da li pristupate elementima izvan dozvoljenog opsega. *Nepoštovanje ove odredbe može biti kažnjeno nepriznavanjem čitavog (inače tačnog) zadatka*!

1. Neki robotski manipulator može se kretati po pravougaonom terenu. Lokacija robota na terenu opisuje se Descartesovim koordinatama $(x, y)$, pri čemu $x$ može biti između $x_{min}$ i $x_{max}$, a $y$ između $y_{min}$ i $y_{max}$ uključivo, pri čemu su $x_{min}, x_{max}, y_{min}$ i $y_{max}$ cijeli brojevi (tipa "`int`") za koje vrijedi $x_{min} < x_{max}$ i $y_{min} < y_{max}$. Prilikom kretanja, robot se može isključivo zaustavljati na lokacijama koje imaju cjelobrojne koordinate, a na početku robot se nalazi tačno u sredini terena (ukoliko su dužina ili širina terena parni brojevi, tako da koordinate tačne sredine terena nisu cjelobrojne, pod "sredinom" treba smatrati onu lokaciju koja bi se dobila zaokruživanjem naniže na cijeli broj koordinata koje odgovaraju tačnoj sredini. U slučaju da robot prilikom kretanja dođe do nekog od rubova terena, a naloženo mu je da se kreće dalje u smjeru koji bi izlazio van terena, robot prosto ostaje na posljednoj validnoj lokaciji koja pripada terenu u smjeru kretanja robota. Robotom se upravlja koristeći odgovarajući skup upravljačkih funkcija. Radi bolje organiziranosti, prototipovi svih ovih upravljačkih funkcija definirani su u posebnom imeniku nazvanom "`Robot`" (time se izbjegava "prljanje" globalnog imenika). Doduše, na predavanjima nije rađeno kako se kreiraju vlastiti imenici, ali ovo je prilika da i to naučimo. Uglavnom, na globalnom nivou u programu nalaziće se deklaracija

```
namespace Robot {
  enum class Pravci {Sjever, Sjeveroistok, Istok, Jugoistok, Jug, Jugozapad, Zapad,
    Sjeverozapad};
  enum class KodoviGresaka {PogresnaKomanda, NedostajeParametar, SuvisanParametar,
    NeispravanParametar};
  enum class Komande {Idi, Rotiraj, Sakrij, Otkrij, PrikaziTeren, Kraj};
  // Ovdje će ići i deklaracije raznih pomoćnih stvari koje će Vam trebati za
  //   potrebe implementacije robota...
  void KreirajTeren(int xmin, int xmax, int ymin, int ymax, int &x, int &o,
    Pravci &orijentacija);
  bool Idi(int &x, int &y, Pravci orijentacija, int korak);
  void Rotiraj(Pravci &orijentacija, int ugao);
  void Sakrij();
  void Otkrij();
  void IspisiPoziciju(int x, int y, Pravci orijentacija);
  void PrikaziTeren();
  void PrijaviGresku(KodoviGresaka kod_greske);
  bool IzvrsiKomandu(Komande komanda, int parametar, int &x, int &y,
    Pravci &orijentacija);
  bool UnosKomande(Komande &komanda, int &parametar, KodoviGresaka &kod_greske);
}
```

Značenja pojedinih stvari koje su deklarirane u ovom imeniku biće data u objašnjenju koje slijedi. Činjenica da su prototipovi svih funkcija navedeni unutar imenika "`Robot`" zahtijevaće navođenje prefiksa "`Robot`" kad god koristimo neku od funkcija iz ovog imenika (npr. "`Robot::Sakrij()`"), osim ukoliko se to ne promijeni pomoću "`using`" deklaracije. Ovo vrijedi i čak kada se radi implementacija ovih funkcija, odnosno čak i tada je prefiks potreban (u suprotnom će se misliti da implementiramo funkciju istog imena, ali u globalnom imeniku), osim ako se implementacija radi unutar samog tijela imenika (i to je moguće, ali nemojte to raditi), jer se unutar tijela imenika podrazumijeva da radimo s tim imenikom. Također, za pristup svim ostalim stvarima definiranim

unutar imenika, neophodan je prefiks (recimo, pobrojanom tipu "`Pravci`" pristupamo pomoću konstrukcije "`Robot::Pravci`", a pobrojanoj konstanti "`Sjever`" tog tipa pomoću konstrukcije "`Robot::Pravci::Sjever`" (umjesto samo "`Pravci::Sjever`"). Jedina iznimka je što se imenik podrazumijeva prilikom implementacije ma koje funkcije koja pripada tom imeniku, pa tada prefiks nije potrebno navoditi. Recimo, ukoliko imamo potrebu da iz funkcije "`PrikaziTeren`" iz imenika "`Robot`" (tj. funkcije "`Robot::PrikaziTeren`") pozovemo funkciju "`IspisiPoziciju`", ne moramo pisati "`Robot::IspisiPoziciju`", nego je dovoljno pisati samo "`IspisiPoziciju`". Slično. za pristup pobrojanom tipu "`Pravci`" ili pobrojanoj konstanti "`Sjever`" iz takve funkcije, nije neophodno pisati "`Robot::Pravci`" odnosno "`Robot::Pravci::Sjever`", nego je dovoljno pisati samo "`Pravci`" odnosno "`Pravci::Sjever`". Zapravo, na nekim mjestima će biti moguće i da se izostavi prefiks "`Robot::`" zahvaljujući jednom pravilu jezika C++ koje se zove ADL (od engl. argument dependent lookup), prema kojem ako argument funkcije pripada nekom imeniku, podrazumijeva se da je i funkcija iz istog tog imenika. Međutim, eksplicitno navođenje prefiksa nigdje neće smetati.

Nakon ovog kratkog uvoda vezanog za kreiranje i korištenje vlastitih imenika, objasnimo čemu služe funkcije čiji su prototipovi navedeni unutar imenika. Prva je funkcija "`KreirajTeren`", čija prva 4 parametra određuju dimenzije terena. Nakon poziva ove funkcije, sve eventualne informacije o ranijem kretanju robota se brišu, kreira se novi teren zadanih dimenzija, i robot se postavlja na početak terena i tako da gleda na sjever (u posljednja 3 parametra treba postaviti početnu poziciju robota). U slučaju da nisu ispunjeni uvjeti $x_{min} < x_{max}$ i $y_{min} < y_{max}$, funkcija treba da baci izuzetak tipa "`range_error`" uz prateći tekst "Nelegalan opseg". Prije nego što se ova funkcija pozove, inicijalno veličina terena treba biti tako da je dozvoljeni raspon od $-10$ do 10 uključivo i po $x$ i po $y$ koordinati.

Funkcija "`Idi`" pomjera robota za broj koraka zadan parametrom "`korak`" u smjeru koji je određen parametrom "`orijentacija`". Ovaj parametar je pobrojanog tipa "`Pravci`", pri čemu su značenja pojedinih pobrojanih konstanti u ovom tipu očigledna. Za teren se pretpostavlja da je orijentiran tako da kretanje na sjever odgovara povećanju $y$ koordinate, uz zadržavanje $x$ koordinate nepromijenjenom. U slučaju "mješovitih" pravaca kao što je npr. jugoistok jedan korak na jugoistok tretira kao da je ekvivalentan jednom koraku na jug i jednom koraku na istok (slično vrijedi i za sve ostale "mješovite" pravce). Trenutna pozicija robota određena je vrijednostima parametara "`x`" i "`y`" na ulazu, a po završetku funkcije isti parametri trebaju sadržavati ažuriranu poziciju. Ukoliko je parametar "`korak`" negativan, kretanje se vrši u smjeru suprotnom od tekuće orijentacije za iznos jednak apsolutnoj vrijednosti parametra. U slučaju da robot prilikom kretanja dođe do ruba terena, a zahtijevano je da ide još dalje, robot treba da se zaustavi upravo na mjestu do kojeg je došao, a funkcija kao rezultat treba da vrati kao rezultat logičku vrijednost "netačno". U suprotnom, odnosno ukoliko je robot prešao svoj put bez nasilnog zaustavljanja, funkcija treba da vrati kao rezultat logičku vrijednost "tačno".

Funkcija "`Rotiraj`" mijenja pravac u kojem robot gleda, odnosno obrće ga nalijevo ili nadesno za ugao od $n \cdot 45°$ u smjeru suprotnom od kazaljke na satu, pri čemu se $n$ zadaje parametrom "`ugao`" (recimo, $n = 2$ odgovara rotaciji za 90°). Parametar $n$ može biti i negativan, ali tada se rotacija vrši u smjeru kazaljke na satu. Parametar "`orijentacija`" predstavlja pravac u kojem robot trenutno gleda, pri čemu funkcija utiče na njegovu vrijednost, tako da će on po završetku funkcije sadržavati novu orijentaciju robota nakon obavljene rotacije.

Funkcija "`Sakrij`" postavlja robota u nevidljivo stanje, a funkcija "`Otkrij`" prebacuje ga u vidljivo stanje. Sve dok je robot vidljiv, prati se (tj. bilježi) svaka lokacija na kojoj se robot nađe tokom svog kretanja. Međutim, dok je robot nevidljiv, njegovo kretanje se ne prati, odnosno on se može kretati kuda god hoće, a da detalji o tome kuda se tačno kretao i gdje je za to vrijeme bio ne budu zabilježeni. Na početku rada programa, robot treba da bude vidljiv.

Funkcija "`IspisiPoziciju`" ispisuje na ekran informacije o poziciju robota u sljedećem formatu:

**`Robot je`** *status*`,` **`nalazi se na poziciji (`** *x*`,` *y*`)` **`i gleda na`** *orijentacija*`.`

"*status*" može biti "vidljiv" ili "nevidljiv", ovisno da li je robot trenutno aktivan ili ne. "*x*" i "*y*" su pozicija robota, koja je određena istoimenim parametrima, dok "*orijentacija*" može biti "sjever", "sjeveroistok", "istok", "jugoistok", "jug", "jugozapad", "zapad" ili "sjeverozapad", u zavisnosti od vrijednosti istoimenog parametra.

Funkcija "PrikaziTeren" prikazuje stanje na terenu na ekranu. Tačan izgled prikaza se može vidjeti u primjeru dijaloga između korisnika i programa, a ukratko izgleda ovako. Teren je omeđen znakovima "#" koji predstavljaju "ogradu" oko terena. Trenutna pozicija robota (bio on vidljiv ili ne) prikazana je znakom "0", dok su sve lokacije na kojima se robot nalazio dok je bio vidljiv označene zvjezdicom (znakom "*"). Sve ostale pozicije (tj. neposjećene, ili one na kojima se robot nalazio *jedino dok je bio nevidljiv*) ostaju prazne.

Funkcija "PrijaviGresku" ima parametar "kod_greske" pobrojanog tipa "KodoviGresaka". Ova funkcija, u zavisnosti od vrijednosti parametra koji joj je proslijeđen, na ekran ispisuje neki od tekstova koji se mogu javiti kao greška u radu sa robotom, u skladu sa sljedećom tabelom (objašnjenje će uslijediti poslije):

| Kôd greške: | Tekst koji treba ispisati: |
|---|---|
| PogresnaKomanda | Nerazumljiva komanda! |
| NedostajeParametar | Komanda trazi parametar koji nije naveden! |
| NeispravanParametar | Parametar komande nije ispravan! |
| SuvisanParametar | Zadan je suvisan parametar nakon komande! |

Prvi parametar funkcije "IzvrsiKomandu" je pobrojanog tipa "Komande". Ova funkcija, u zavisnosti od vrijednosti tog parametra, izvršava zadanu komandu, pozivom odgovarajuće funkcije za izvršenje komande (osim ukoliko je komanda "Kraj"; tada funkcija ne radi ništa). Parametar "parametar" koristi se samo ukoliko je komanda "Idi" ili "Rotiraj", i on tada predstavlja vrijednost koja se prosljeđuje kao posljednji parametar istoimenim funkcijama (tj. funkcijama "Idi" odnosno "Rotiraj"). U svim ostalim slučajevima, vrijednost ovog parametra se ignorira. Preostali parametri funkcije "IzvrsiKomandu" predstavljaju informaciju o poziciji i orijentaciji robota. Kao rezultat, ova funkcija uvijek vraća logičku vrijednost "tačno", osim u slučaju kada je komanda bila "Idi", a kretanje nije uspješno izvršeno (pokušaj izlaska robota van terena).

Komunikacija između korisnika i robota vrši se posredstvom funkcije "UnosKomande". Ova funkcija očekuje od korisnika da zada unos komande putem tastature. Legalne komande su sljedeće (u komandama "S+" i "S-" znakovi "+" i "-" su dio komande a ne parametar):

| Komanda: | Značenje |
|---|---|
| **I** *korak* | Pomjera robota za navedeni broj koraka |
| **R** *ugao* | Rotira robota za navedeni ugao, pri čemu je jedinica mjere 45° |
| **S+** | Sakriva robota, tj. čini ga nevidljivim |
| **S–** | Otkriva robota, tj. čini ga vidljivim |
| **T** | Prikazuje stanje na terenu |
| **K** | Završetak rada programa |

Razmaci ispred i iza komande su dozvoljeni, ali bilo kakav neočekivani znak (osim razmaka) nakon komande tretira se kao suvišan parametar. Komande "I" odnosno "R" su praćene cijelim brojem (koji može biti i negativan) koji predstavlja broj koraka koji će robot napraviti, odnosno ugao rotacije. Razmaci između komande i broja su dozvoljeni (ali ne i obavezni), pri čemu se bilo šta što ne predstavlja ispravan cijeli broj (uključujući suvišne znakove iza broja) tretira kao neispravan parametar. U slučaju da se prepozna ispravna komanda, funkcija smješta njen kôd u parametar "komanda", eventualni parametar komande smješta se u parametar "parametar" (u slučaju da komanda nema parametra, vrijednost parametra "parametar" je nedefinirana), a funkcija vraća kao rezultat logičku vrijednost "**true**" kao signal da je komanda prepoznata. Parametar "kod_greske" tada je nedefiniran. U suprotnom, ukoliko nije prepoznata ispravna komanda, kôd greške se smješta u parametar "kod_greske", parametri "komanda" i "parametar" su nedefinirani, a funkcija vraća kao rezultat logičku vrijednost "**false**" kao signal da nije prepoznata ispravna komanda. Eventualni razmaci ispred komande kao i nakon komande su dozvoljeni i ne trebaju uzrokovati prikavu greške. Potrebno je predvidjeti sve što bi korisnik eventualno mogao unijeti (funkcija kao i program koji je koristi ne smije da "crkne" šta god korisnik unio).

Napisane funkcije treba demonstrirati u glavnom programu u kojem će se prvo tražiti da se s tastature unesu informacije o željenom formatu terena. Ukoliko su informacije neispravne, treba ispisati tekst "Nije moguce kreirati takav teren!" i prekinuti s radom. U suprotnom, treba kreirati

teren i postaviti robota na početnu poziciju (pozivom funkcije "`KreirajTeren`"), nakon čega program treba u petlji zahtijevati unos komande pozivom funkcije "`UnosKomande`". Po unosu komande, ona se ili izvršava (pozivom funkcije "`IzvrsiKomandu`") ili se prijavljuje greška (pozivom funkcije "`PrijaviGresku`"). Petlja se prekida nakon što se zada komanda "`Kraj`". Tada program ispisuje poruku "Dovidjenja!" i završava sa radom. Slijedi primjer kako može izgledati dijalog između korisnika i programa:

```
Unesite dimenzije terena (xmin xmax ymin ymax): -30 30 -10 10
Robot je vidljiv, nalazi se na poziciji (0,0) i gleda na sjever.
Unesite komandu: I7
Robot je vidljiv, nalazi se na poziciji (0,7) i gleda na sjever.
Unesite komandu: R1
Robot je vidljiv, nalazi se na poziciji (0,7) i gleda na sjeverozapad.
Unesite komandu: I20
Robot je pokusao napustiti teren!
Robot je vidljiv, nalazi se na poziciji (-3,10) i gleda na sjeverozapad.
Unesite komandu: S+
Robot je nevidljiv, nalazi se na poziciji (-3,10) i gleda na sjeverozapad.
Unesite komandu: R-3
Robot je nevidljiv, nalazi se na poziciji (-3,10) i gleda na istok.
Unesite komandu: I25
Robot je nevidljiv, nalazi se na poziciji (22,10) i gleda na istok.
Unesite komandu: R5
Robot je nevidljiv, nalazi se na poziciji (22,10) i gleda na jugozapad.
Unesite komandu: I-3
Robot je pokusao napustiti teren!
Robot je nevidljiv, nalazi se na poziciji (22,10) i gleda na jugozapad.
Unesite komandu: I7
Robot je nevidljiv, nalazi se na poziciji (15,3) i gleda na jugozapad.
Unesite komandu: S-
Robot je vidljiv, nalazi se na poziciji (15,3) i gleda na jugozapad.
Unesite komandu: R11
Robot je vidljiv, nalazi se na poziciji (15,3) i gleda na istok.
Unesite komandu: I-35
Robot je vidljiv, nalazi se na poziciji (-20,3) i gleda na istok.
Unesite komandu: R-1
Robot je vidljiv, nalazi se na poziciji (-20,3) i gleda na jugoistok.
Unesite komandu: I10
Robot je vidljiv, nalazi se na poziciji (-10,-7) i gleda na jugoistok.
Unesite komandu: T
##################################################################
#                            *                                   #
#                             *                                  #
#                              *                                 #
#                               *                                #
#                               *                                #
#                               *                                #
#                               *                                #
#          ************************************                  #
#           *                   *                                #
#            *                  *                                #
#             *                 *                                #
#              *                                                 #
#               *                                                #
#                *                                               #
#                 *                                              #
#                  *                                             #
#                   *                                            #
#                    *                                           #
#                     O                                          #
#                                                                #
#                                                                #
#                                                                #
##################################################################
Unesite komandu: R-3
Robot je vidljiv, nalazi se na poziciji (-10,-7) i gleda na zapad.
Unesi komandu: K
Dovidjenja!
```

Slijedi još jedan primjer kako može izgledati dijalog između korisnika i programa, u kojem je pretpostavljeno da su granice terena asimetrične, odnosno da ne vrijedi $x_{min} = -x_{max}$, niti

$y_{min} = -y_{max}$. Također dijalog iz sljedećeg primjera prikazuje razne primjere ispravnih odnosno neispravnih unosa, te kako treba tačno reagirati u slučaju neispravnih unosa:

```
Unesite dimenzije terena (xmin xmax ymin ymax): -30 10 -6 10
Robot je vidljiv, nalazi se na poziciji (-10,2) i gleda na sjever.
Unesite komandu: R-2
Robot je vidljiv, nalazi se na poziciji (-10,2) i gleda na istok.
Unesite komandu: I
Komanda trazi parametar koji nije naveden!
Unesite komandu: I  5
Robot je vidljiv, nalazi se na poziciji (-5,2) i gleda na istok.
Unesite komandu: S+
Robot je nevidljiv, nalazi se na poziciji (-5,2) i gleda na istok.
Unesite komandu:    R -1
Robot je nevidljiv, nalazi se na poziciji (-5,2) i gleda na jugoistok.
Unesite komandu:    I3
Robot je nevidljiv, nalazi se na poziciji (-2,-1) i gleda na jugoistok.
Unesite komandu: R7
Robot je nevidljiv, nalazi se na poziciji (-2,-1) i gleda na jug.
Unesite komandu: M
Nerazumljiva komanda!
Unesite komandu: S-
Robot je vidljiv, nalazi se na poziciji (-2,-1) i gleda na jug.
Unesite komandu: IXY2
Parametar komande nije ispravan!
Unesite komandu: R-2
Robot je vidljiv, nalazi se na poziciji (-2,-1) i gleda na zapad.
Unesite komandu: I2XY
Parametar komande nije ispravan!
Unesite komandu: I0
Robot je vidljiv, nalazi se na poziciji (-2,-1) i gleda na zapad.
Unesite komandu: I10
Robot je vidljiv, nalazi se na poziciji (-12,-1) i gleda na zapad.
Unesite komandu: T2
Zadan je suvisan parametar nakon komande!
Unesite komandu: R3
Robot je vidljiv, nalazi se na poziciji (-12,-1) i gleda na jugoistok.
Unesite komandu: I8
Robot je pokusao napustiti teren!
Robot je vidljiv, nalazi se na poziciji (-7,-6) i gleda na jugoistok.
Unesite komandu: T
##########################################
#                                        #
#                                        #
#                                        #
#                                        #
#                                        #
#                                        #
#                                        #
#                                        #
#                    ******              #
#                                        #
#                                        #
#                  ***********           #
#                   *                    #
#                    *                   #
#                     *                  #
#                      *                 #
#                       O                #
##########################################
Unesite komandu: R4
Robot je vidljiv, nalazi se na poziciji (-7,-6) i gleda na sjeverozapad.
Unesite komandu: R0
Robot je vidljiv, nalazi se na poziciji (-7,-6) i gleda na sjeverozapad.
Unesite komandu: I7
Robot je vidljiv, nalazi se na poziciji (-14,1) i gleda na sjeverozapad.
Unesite komandu: R5
Robot je vidljiv, nalazi se na poziciji (-14,1) i gleda na istok.
Unesite komandu: KK
Zadan je suvisan parametar nakon komande!
Unesite komandu: K
Dovidjenja!
```

VAŽNA NAPOMENA: Jasno je da će Vam za realizaciju ovog zadatka biti potrebno i korištenje globalnih promjenljivih. Međutim, sve globalne promjenljive koje Vam eventualno budu potrebne čuvajte isključivo u imeniku "Robot". Međutim, informacije o poziciji i orijentaciji robota *ne smiju se čuvati u globalnim promjenljivim*, nego se te informacije moraju razmjenjivati između funkcija putem prenosa parametara. Također, u programu *nije dozvoljeno koristiti unos sa tastature u promjenljive tipa string ili niz znakova*, nego sva procesiranja ulaza treba vršiti direktnim čitanjima iz spremnika ulaznog toka. Nepoštovanje ovih ograničenja biće kažnjeno davanjem 0 poena na čitav zadatak (koji eventualno može biti i tačan)!

2. Neka su date dvije matrice $\boldsymbol{A}$ i $\boldsymbol{B}$, formata $m \times n$ i $p \times q$ respektivno:

$$\boldsymbol{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix} \qquad \boldsymbol{B} = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1q} \\ b_{21} & b_{22} & \dots & b_{2q} \\ \vdots & \vdots & & \vdots \\ b_{p1} & b_{p2} & \dots & b_{pq} \end{pmatrix}$$

Pod *Kroneckerovim* (ili *tenzorskim*) *proizvodom* ovih matrica smatra se matrica $\boldsymbol{A} \otimes \boldsymbol{B}$ formata $mp \times nq$ definirana kao

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{pmatrix} a_{11}b_{11} & a_{11}b_{12} & \dots & a_{11}b_{1q} & a_{12}b_{11} & a_{12}b_{12} & \dots & a_{12}b_{1q} & \dots & a_{1n}b_{11} & a_{1n}b_{12} & \dots & a_{1n}b_{1q} \\ a_{11}b_{21} & a_{11}b_{22} & \dots & a_{11}b_{2q} & a_{12}b_{21} & a_{12}b_{22} & \dots & a_{12}b_{2q} & \dots & a_{1n}b_{21} & a_{1n}b_{22} & \dots & a_{1n}b_{2q} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & & \vdots & \vdots & & \vdots \\ a_{11}b_{p1} & a_{11}b_{p2} & \dots & a_{11}b_{pq} & a_{12}b_{p1} & a_{12}b_{p2} & \dots & a_{12}b_{pq} & \dots & a_{1n}b_{p1} & a_{1n}b_{p2} & \dots & a_{1n}b_{pq} \\ a_{21}b_{11} & a_{21}b_{12} & \dots & a_{21}b_{1q} & a_{22}b_{11} & a_{22}b_{12} & \dots & a_{22}b_{1q} & \dots & a_{2n}b_{11} & a_{2n}b_{12} & \dots & a_{2n}b_{1q} \\ a_{21}b_{21} & a_{21}b_{22} & \dots & a_{21}b_{2q} & a_{22}b_{21} & a_{22}b_{22} & \dots & a_{22}b_{2q} & \dots & a_{2n}b_{21} & a_{2n}b_{22} & \dots & a_{2n}b_{2q} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & & \vdots & \vdots & & \vdots \\ a_{21}b_{p1} & a_{21}b_{p2} & \dots & a_{21}b_{pq} & a_{22}b_{p1} & a_{22}b_{p2} & & a_{22}b_{pq} & & a_{2n}b_{p1} & a_{2n}b_{p2} & & a_{2n}b_{pq} \\ & & & & & & & & & & & & \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & & \vdots & \vdots & & \vdots \\ & & & & & & & & & & & & \\ a_{m1}b_{11} & a_{m1}b_{12} & \dots & a_{m1}b_{1q} & a_{m2}b_{11} & a_{m2}b_{12} & \dots & a_{m2}b_{1q} & \dots & a_{mn}b_{11} & a_{mn}b_{12} & \dots & a_{mn}b_{1q} \\ a_{m1}b_{21} & a_{m1}b_{22} & \dots & a_{m1}b_{2q} & a_{m2}b_{21} & a_{m2}b_{22} & \dots & a_{m2}b_{2q} & \dots & a_{mn}b_{21} & a_{mn}b_{22} & \dots & a_{mn}b_{2q} \\ \vdots & \vdots & & \vdots & \vdots & \vdots & & \vdots & & \vdots & \vdots & & \vdots \\ a_{m1}b_{p1} & a_{m1}b_{p2} & \dots & a_{m1}b_{pq} & a_{m2}b_{p1} & a_{m2}b_{p2} & \dots & a_{m2}b_{pq} & \dots & a_{mn}b_{p1} & a_{mn}b_{p2} & \dots & a_{mn}b_{pq} \end{pmatrix}$$

Na primjer, ukoliko su matrice $\boldsymbol{A}$ i $\boldsymbol{B}$ date kao

$$\boldsymbol{A} = \begin{pmatrix} 3 & 5 & 2 \\ 4 & 0 & -1 \end{pmatrix} \qquad \boldsymbol{B} = \begin{pmatrix} 1 & 4 & -2 & 3 \\ 0 & 5 & 4 & 1 \\ 2 & 0 & 0 & 3 \end{pmatrix}$$

njihov Kroneckerov proizvod glasi

$$\boldsymbol{A} \otimes \boldsymbol{B} = \begin{pmatrix} 3 & 12 & -6 & 9 & 5 & 20 & -10 & 15 & 2 & 8 & -4 & 6 \\ 0 & 15 & 12 & 3 & 0 & 25 & 20 & 5 & 0 & 10 & 8 & 2 \\ 6 & 0 & 0 & 9 & 10 & 0 & 0 & 15 & 4 & 0 & 0 & 6 \\ 4 & 16 & -8 & 12 & 0 & 0 & 0 & 0 & -1 & -4 & 2 & -3 \\ 0 & 20 & 16 & 4 & 0 & 0 & 0 & 0 & 0 & -5 & -4 & -1 \\ 8 & 0 & 0 & 12 & 0 & 0 & 0 & 0 & -2 & 0 & 0 & -3 \end{pmatrix}$$

Dalje, neka je data neka funkcija $f$ s dva parametra, pri čemu je prvi parametar onog tipa kakav je tip elemenata matrice $\boldsymbol{A}$, a drugi parametar onog tipa kakav je tip elemenata matrice $\boldsymbol{B}$. Tvorevinu sličnu Kroneckerovom proizvodu matrica $\boldsymbol{A}$ i $\boldsymbol{B}$, ali u kojoj su produkti elemenata zamijenjeni rezultatima poziva funkcije $f$, tačnije u kojoj umjesto produkata oblika $a_{ij}b_{kl}$ imamo članove oblika $f(a_{ij}, b_{kl})$ nazivamo generalizirani Kroneckerov (ili tenzorski) produkt matrica $\boldsymbol{A}$ i $\boldsymbol{B}$ i označavamo ga s $\boldsymbol{A} \otimes_f \boldsymbol{B}$. Na primjer, ukoliko su $\boldsymbol{A}$ i $\boldsymbol{B}$ iste matrice kao u prethodnom primjeru i ukoliko je $f(x, y) = x^2 + y$, tada imamo

$$\boldsymbol{A} \otimes_f \boldsymbol{B} = \begin{pmatrix} 10 & 13 & 7 & 12 & 26 & 29 & 23 & 28 & 5 & 8 & 2 & 7 \\ 9 & 14 & 13 & 10 & 25 & 30 & 29 & 26 & 4 & 9 & 8 & 5 \\ 11 & 9 & 9 & 12 & 27 & 25 & 25 & 28 & 6 & 4 & 4 & 7 \\ 17 & 20 & 14 & 19 & 1 & 4 & -2 & 3 & 2 & 5 & -1 & 4 \\ 16 & 21 & 20 & 17 & 0 & 5 & 4 & 1 & 1 & 6 & 5 & 2 \\ 18 & 16 & 16 & 19 & 2 & 0 & 0 & 3 & 3 & 1 & 1 & 4 \end{pmatrix}$$

Napišite generičku funkciju "`GeneraliziraniKroneckerovProizvod`" koja računa generalizirani matrčini proizvod matrica koje koje joj se prenose kao parametri. Matrice su organizirani kao proizvoljni dvodimenzionalni kontejneri (to mogu biti recimo vektori vektôrā, dekovi modernih nizova, pa čak i nizovi nizova, odnosno klasične C-ovske matrice), za koje se jedino pretpostavlja da su takvi da podržavaju indeksiranje parom indeksa (koji predstavljaju indekse redova odnosno kolona). Tip elemenata u prvom i drugom kontejneru može se razlikovati. Treći parametar je funkcija $f$ koja ima dva parametra koji su onakvog tipa kakvog su tipa elementi prve odnosno druge matrice, a njen rezultat može biti ma kakvog tipa. Pri tome, da biste izbjegli vrlo složenu deklaraciju ovog funkcijskog parametra kakva bi bila neophodna prema prethodnoj specifikaciji, pustite da se tip ovog funkcijskog parametra određuje putem potpune dedukcije (jedini nedostatak ovog pristupa je što će se instantacija funkcije pokušati šta god da pošaljemo ovoj funkciji kao treći parametar, koja će onda eventualno pasti u slučaju da taj parametar nisu validni). Kao rezultat, funkcija treba da vrati matricu organiziranu kao vektor vektôrā onakvog tipa elemenata kakav vraća funkcija $f$, a koja predstavlja generalizirani Kroneckerov proizvod matrica koje su prenesene kao prva dva parametra. U slučaju da neki od prva dva parametra nemaju ispravnu formu matrice, u smislu da im različiti redovi nemaju isti broj elemenata, funkcija treba da baci izuzetak tipa "`domain_error`" uz jedan od pratećih tekstova "Prvi parametar nema formu matrice", "Drugi parametar nema formu matrice", ili "Parametri nemaju formu matrice", ovisno od toga o čemu je zaista riječ. U slučaju da se tokom računanja desi da funkcija $f$ baci bilo kakav izuzetak, funkcija "`GeneraliziraniKroneckerovProizvod`" treba baciti izuzetak tipa "`runtime_error`" uz prateći tekst "Neocekivani problemi pri racunanju".

Napisanu funkciju demonstrirajte u testnom programu u kojem se sa tastature prvo traži unos dimenzija prve matrice, zatim elementi prve matrice, zatim dimenzije druge matrice, te konačno elementi druge matrice. Elementi obje matrice trebaju biti realni brojevi, pri čemu je prva organizirana kao vektor dekova, a druga kao dek vektora. Program nakon toga treba da pozivom napisane funkcije izračuna klasični Kroneckerov proizvod unesenih matrica (za koji vrijedi da je $f(x,y) = xy$) i da ga ispiše na ekranu. Ispis treba vršiti red po red, pri čemu za ispis elemenata treba fiksirati širinu na iznos koji je za 1 veći od širine najvećeg elementa matrice, pri čemu se pod širinom broja podrazumijeva broj njegovih cifara, eventualno uvećan za 1 ukoliko je broj negativan. Ispis treba biti poravnat udesno u rezervirani prostor. Dijalog između korisnika i programa treba izgledati poput sljedećeg:

```
Unesite dimenzije prve matrice: 2 2
Unesite elemente prve matrice: 3 -1 0 5
Unesite dimenzije druge matrice: 2 3
Unesite elemente druge matrice: 4 3 15 0 -5 2
Njihov Kroneckerov proizvod glasi:
  12   9  45  -4  -3 -15
   0 -15   6   0   5  -2
   0   0   0  20  15  75
   0   0   0   0 -25  10
```

NAPOMENA 1: Da osmislite algoritam, prvo pokušajte napisati funkciju uz pretpostavku da su parametri isključivo matrice organizirane kao vektori vektôrā. Tek kad Vam funkcija ispravno proradi uz takve pretpostavke, uopćite funkciju da može prihvatati i općenitije tipove podataka. Ukoliko ne uspijete napraviti funkciju u svojoj punoj općenitosti, probajte je napraviti da radi za što više specijalnih slučajeva. Nešto ćete poena dobiti i za to.

NAPOMENA 2: Rečeno je da parametri mogu biti čak i nizovi nizova. Da bi funkcija radila ispravno i u tom slučaju, morate voditi računa o dvije stvari. Prvo, morate spriječiti da se nizovi prilikom slanja u funkciju automatski konvertiraju u pokazivač na prvi element niza. Ukoliko se to ne spriječi, funkcija neće raditi korektno za takve parametre. Drugo, trebaće Vam informacije o

broju redova i kolona matrica. To je lako dobiti za većinu dvodimenzionalnih kontejnera, ali nije recimo za matrice organizirane kao nizovi nizova, nizovi vektôrā, itd. s obzirom da nizovi (misli se na klasične C-ovske nizove) ne podržavaju funkciju "`size`". Međutim, nizovi podržavaju funkcije "`begin`" i "`end`", uz ispravnu sintaksu (pod uvjetom da nisu degenerirali u pokazivače). Razmislite kako možete iskoristiti ove funkcije da saznate broj elemenata niza.

NAPOMENA 3: Rečeno je da kontejneri podržavaju indeksiranje. To nažalost ne znači da je nužno podržana funkcija "`at`" (recimo, ona nije podržana za klasične nizove nizova, odnosno C-ovske matrice), nego samo pristup elementima putem uglastih zagrada. To znači da je u ovom zadatku dozvoljeno indeksirati kontejner putem uglastih zagrada. Međutim, nije uopće teško u potpunosti izbjeći indeksiranje, tako da se ni ova pretpostavka ne mora koristiti (mnogo o tome možete naučiti analizom rješenja zadataka 9, 10, 11 i 12 iz Riješenih zadataka uz Predavanje 6). Ukoliko hoćete da se baš uvježbate za rješavanje ovakve vrste problema, probajte pri rješavanju ovog zadatka u potpunosti izbjeći indeksiranje.

3. Napišite generičku funkciju "`UnijaBlokova`" koja ima pet parametara $p_1$, $p_2$, $p_3$, $p_4$ i $p_5$, koji mogu biti pokazivači ili iteratori. Parametri $p_1$ i $p_2$ omeđuju jedan blok elemenata ($p_1$ pokazuje na početak bloka a $p_2$ tačno iza njegovog kraja), dok $p_3$ i $p_4$ omeđuju drugi blok elemenata. Funkcija treba da u blok na čiji početak pokazuje $p_5$ upiše elemente koji se nalaze bilo u jednom bloku, bilo u drugom bloku, bilo i u jednom i drugom bloku (tj. da napravi uniju ova dva bloka). Pri tome, nijedan element ne treba upisivati više puta u odredišni blok, odnosno odredišni blok ne smije sadržavati duplikate. Redoslijed elemenata u odredišnom bloku treba biti takav da se prvo smještaju svi elementi iz prvog bloka (neovisno od toga ima li ih u drugom bloku ili ne), naravno preskačući duplikate (tj. elemente koji su već smješteni u odredišni blok, a zatim se smještaju elementi iz drugog bloka koji nisu prije toga bili smješteni u odredišni blok (bilo zbog toga što se nalaze i u prvom bloku, bilo zbog toga što je već ranije smješten identičan element iz drugog bloka). Na primjer, ukoliko prvi blok sadrži elemente 5, 2, 7, 4, 2, 6, 1, 7, 3 i 4, 2 i 5 a drugi blok elemente 2, 9, 0, 6, 0, 4, 8, 3, 2, 5 i 7, u odredišni blok treba upisati elemente 5, 2, 7, 4, 6, 1, 3, 9, 0 i 8. Kao rezultat, funkcija treba da vrati pokazivač ili iterator (zavisno da li je $p_5$ pokazivač ili iterator) koji pokazuje tačno iza kraja bloka u koji sadrži nađenu uniju blokova. Parametri $p_1$ i $p_2$ trebaju biti istog tipa. Isto vrijedi i za $p_3$ i $p_4$. Međutim, tipovi za $p_1$ i $p_3$ nisu nužno isti (recimo, $p_1$ može biti pokazivač, a $p_3$ iterator). Tip $p_5$ može biti posve neovisan od tipa ostalih parametara. Pri tome su jedine dozvoljene operacije koje smjete koristiti s pokazivačima i/ili iteratora dereferenciranje ("`*`"), dodjela ("`=`"), inkrementiranje ("`++`"), te poređenje na jednakost ("`==`") odnosno različitost ("`!=`"). Razlog za ova ograničenja je da bi se funkcija mogla koristiti i s kontejnerskim tipovima čiji iteratori nemaju punu snagu kakvu imaju pokazivači (tj. čiji iteratori nisu iteratori s direktnim pristupom).

   Napisanu funkciju demonstrirajte u testnom programu koji traži da se sa tastature unesu elementi jednog niza realnih brojeva i jednog deka cijelih brojeva (elementi niza i deka unose se sa tastature, pri čemu broj elemenata niza neće biti veći od 100) a koji zatim poziva napisanu funkciju da pronađe uniju elemente koji se javljaju i u nizu i u deku i smjesti ih u jedan novi vektor realnih brojeva. Potrebno je iskoristiti povratnu vrijednost funkcije da se veličina vektora na kraju namjesti da bude tačno onolika koliko ima nađenih zajedničkih elemenata, nakon čega se nađeni elementi trebaju ispisati na ekran (pri čemu trebate voditi računa da će autotestovi napisanu funkciju možda pozivati i nad drugim kontejnerskim tipovima podataka). Dijalog između korisnika i programa treba izgledati kao na sljedećoj slici:

```
Unesite broj elemenata niza (max. 100): 12
Unesite elemente niza: 5 2 7 4 2 6 1 7 3 4 2 5
Unesite broj elemenata deka: 11
Unesite elemente deka: 2 9 0 6 0 4 8 3 2 5 7
Njihova unija glasi: 5 2 7 4 6 1 3 9 0 8
```

4. Nazovimo jedan red matrice boljim od nekog drugog reda ukoliko je njegov najveći element veći od najvećeg elementa tog drugog reda. Napravite generičku funkciju "`SortirajPoDobrotiRedova`" koja kao parametar kao jedini parametar neku matricu (koja može biti i grbava) organiziranu kao vektor vektôrā proizvoljnog tipa elemenata, za koje se jedino pretpostavlja da se mogu porediti po veličini Funkcija treba da transformira tu matricu tako da ona postane sortirana po "dobroti" redova u rastući poredak, u smislu da nakon sortiranja njen prvi red bude onaj red koji

je "najbolji" u prethodno opisanom smislu, zatim slijedi sljedeći red po "dobroti", i tako dalje sve do posljednjeg reda, koji treba da bude "najlošiji". Ukoliko dva reda imaju istu "dobrotu", tada prije treba da dođe onaj red koji je veći (tj. koji dolazi kasnije) prema leksikografskom poretku. Funkcija ne vraća nikakav rezultat, nego samo modificira matricu koja joj se šalje kao parametar.

Traženu funkciju obavezno treba realizirati uz pomoć funkcije "`sort`" iz biblioteke s zaglavljem "`algorithm`", kojoj treba proslijediti odgovarajuću funkciju kriterija, koju treba izvesti kao imenovanu funkciju nazvanu "`Kriterij`". Pri tome, ta funkcija kriterija ne smije koristiti nikakve petlje (ali smije pozivati druge funkcije iz iste biblioteke).

Napisanu funkciju demonstrirajte u testnom programu u kojem se elementi grbave matrice cijelih brojeva unose red po red, pri čemu oznaku kraja reda predstavlja bilo šta što nije broj (recimo "*"). Unos se završava kada se odmah na početku reda unese nešto što nije broj. Nakon obavljenog unosa, program treba da sortira matricu na opisani način, i da ispiše njene elemente nakon obavljenog sortiranja. Konačno, nakon ispisa, program treba da traži unos elemenata neke sekvence cijelih brojeva, nakon čega postupkom binarne pretrage testira da li se unesena sekvenca pojavljuje kao neki od redova ranije unesene matrice. Ovisno od rezultata testiranja, program treba da u slučaju uspješne pretrage ispiše rečenicu "Trazena sekvenca se nalazi u *i*. redu (nakon sortiranja)", pri čemu je $i$ red matrice koji je identičan unesenoj sekvenci (nakon obavljenog sortiranja), odnosno rečenicu "Trazena sekvenca se ne nalazi u matrici" u slučaju neuspješne pretrage. Ukoliko ima više redova koji su identični unesenoj sekvenci, treba prijaviti prvi od njih. Za pretragu koristite isključivo funkciju "`lower_bound`" iz biblioteke sa zaglavljem "`algorithm`" (i nijednu drugu). Dijalog između korisnika i programa treba da izgleda poput sljedećeg:

```
Unesite elemente (* za kraj reda, * na pocetku reda za kraj unosa):
2 5 1 6 7 *
9 8 9 *
3 3 2 3 *
4 5 1 1 7 3 2 *
*

Matrica nakon sortiranja:
9 8 9
4 5 1 1 7 3 2
2 5 1 6 7
3 3 2 3

Unesite elemente sekvence koja se trazi (* za kraj reda): 2 5 1 6 7 *
Trazena sekvenca se nalazi u 3. redu (nakon sortiranja)
```

5. Date su dvije sekvence $a_1, a_2, \ldots, a_n$ i $b_1, b_2, \ldots, b_n$ iste dužine. Pretpostavlja se da se elementi prve sekvence mogu množiti s elementima druge sekvence i da je pri tome množenje komutativno, tj. da za ma koje $a_i$ i $b_j$ vrijedi $a_i\, b_j = b_j\, a_i$. Potrebno je napraviti trougaonu tablicu množenja, koja će sadržavati proizvode $a_i\, b_j$ za $i = 1, 2, \ldots, n$ i $j = 1, 2, \ldots, i$, kao na sljedećoj slici:

$$
\begin{array}{llllll}
a_1\, b_1 & & & & & \\
a_2\, b_1 & a_2\, b_2 & & & & \\
a_3\, b_1 & a_3\, b_2 & a_3\, b_2 & & & \\
\ldots & \ldots & \ldots & & & \\
a_{n-1}\, b_1 & a_{n-1}\, b_2 & a_{n-1}\, b_3 & \ldots & a_{n-1}\, b_{n-1} & \\
a_n\, b_1 & a_n\, b_2 & a_n\, b_3 & \ldots & a_n\, b_{n-1} & a_n\, b_n
\end{array}
$$

Vaš zadatak je da napravite generičku funkciju nazvanu "`KreirajTablicuMnozenja`" koja kreira elemente ovakve tablice. Prva dva parametra funkcije su proizvoljni kontejneri (vektori, dekovi, itd. pa čak i obični nizovi), od kojih prvi sadrži sekvencu $a_1, a_2, \ldots, a_n$, dok drugi sadrži sekvencu $b_1, b_2, \ldots, b_n$. Ukoliko ova dva kontejnera nemaju isti broj elemenata, treba baciti izuzetak tipa "`range_error`" uz prateći tekst "Kontejneri nisu iste duzine". Kontejneri se mogu razlikovati po tipu (naravno, to povlači i da elementi ove dvije sekvence ne moraju nužno biti istog tipa). Za kontejnere se ne pretpostavlja čak ni da podržavaju indeksiranje (na Predavanju 7_b vidjećemo da ima i takvih kontejnera), nego samo da podržavaju funkcije "`begin`" i "`end`" koje vraćaju pokazivač ili iterator na njihov prvi element, odnosno tačno iza kraja posljednjeg elementa. Pri tome, da bi funkcija bila što univerzalnija, za te iteratore nemojte pretpostavljati da podržavaju

nikakve druge operacije od onih operacija koje svaki iterator mora da podržava (nisu svi iteratori jednako moćni s aspekta operacija koje podržavaju, kao što će biti objašnjeno kasnije na Predavanju 7_b), a to su operacije dereferenciranja ("`*`"), dodjele ("`=`"), inkrementiranja ("`++`"), te poređenja na jednakost i različitost ("`==`" i "`!=`"). Vodite računa da će funkcija biti testirana i sa kontejnerima čiji iteratori ne podržavaju ništa više od ovih operacija (vidjećete kasnije da postoje i takvi kontejnerski tipovi). Ovim ograničenjem na korištene operatore znatno se proširuje univerzalnost funkcije, jer će se moći primijeniti i s takvim kontejnerima.

Funkcija treba dinamički alocirati grbavu matricu čiji prvi red ima jedan element, drugi red dva elementa, itd. i popuniti je rezultatima množenja. Kreiranje treba obaviti postupkom *kontinualne alokacije*. Tip elemenata alocirane matrice treba da bude onakav kakvog je tipa rezultat množenja elemenata iz prve i druge sekvence (recimo, ako prva sekvenca sadrži realne, a druga kompleksne brojeve, elementi matrice trebaju biti kompleksnog tipa). Kao rezultat, funkcija treba da vrati dvojni pokazivač pomoću kojeg se može pristupiti elementima kreirane matrice. Ukoliko se dogodi da alokacija ne uspije zbog nedovoljne količine raspoložive memorije, funkcija treba baciti izuzetak tipa "`range_error`" uz prateći tekst "Nema dovoljno memorije". Pri tome, treba paziti da ni u kom slučaju ne smije doći do curenja memorije.

Kao što je već rečeno, polazimo od pretpostavke da je množenje komutativno. Međutim, ukoliko funkcija ustanovi da to nije ispunjeno, odnosno da za makar jedan par indeksa $i$ i $j$ vrijedi da je $b_j\, a_i \neq a_i\, b_j$ (postoje tipovi podataka za koje u općem slučaju nije komutativno), funkcija treba baciti izuzetak tipa "`logic_error`" uz prateći tekst "Nije ispunjena pretpostavka o komutativnosti".

Napisanu funkciju demonstrirajte u kratkom testnom programu, koji traži unos dužine sekvenci kao i njihove elemente, od kojih prvu treba smjestiti u vektor a drugu u dek realnih brojeva, a koji zatim računa i ispisuje na ekran elemente kreirane tablice sabiranja, te na kraju oslobađa svu alociranu memoriju. Elementi svakog od redova se ispisuju razdvojeni po jednim razmakom. Dijalog između korisnika i programa treba da izgleda poput sljedećeg:

```
Duzina sekvenci: 4
Elementi prve sekvence: 5 2 8 7
Elementi druge sekvence: 1 3 6 2
Tablica množenja:
5
2 6
8 24 48
7 21 42 14
```

NAPOMENA: Sve napomene koje vrijede za Zadatak 2, vrijede i za ovaj zadatak. Posebno, ukoliko ne znate napisati funkciju da bude onoliko općenita koliko se traži u zadatku, napravite je da radi za što više specijalnih slučajeva. Dobićete nešto bodova i na takvu izvedbu.