Zadaća 2.

Ova zadaća nosi ukupno 5 poena, od kojih prvi zadatak nosi 1,4 poena, a ostali po 0,9 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 300 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 nedjelja, 24. IV 2022. 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. Za upravljanje nekim robotskim manipulatorom koristi se skup upravljačkih funkcija čiji su prototipovi sljedeći:

```
void AktivirajRobota();
void DeaktivirajRobota();
void Rotiraj(Pravci &orijentacija, Smjer na_koju_stranu);
void Idi(int &x, int &y, Pravci orijentacija, int korak);
void IspisiPoziciju(int x, int y, Pravci orijentacija);
void PrijaviGresku(KodoviGresaka kod_greske);
void IzvrsiKomandu(Komande komanda, int parametar, int &x, int &y,
  Pravci &orijentacija);
```

   Funkcije "`AktivirajRobota`" i "`DeaktivirajRobota`" aktiviraju odnosno deaktiviraju robota. Robot izvršava komande samo kada je aktivan. U suprotnom, dok je deaktiviran, robot ignorira svaku komandu koja mu se uputi, odnosno prihvata komandu, ali je ne izvršava.

   Funkcija "`Rotiraj`" mijenja pravac u kojem robot gleda, odnosno obrće ga nalijevo ili nadesno za 45° ovisno od vrijednosti parametra "`na_koju_stranu`". Ovaj parametar može imati samo dvije vrijednosti "`Nalijevo`" ili "`Nadesno`". Njegov tip je "`Smjer`", što predstavlja pobrojani tip koji je deklariran (na globalnom nivou) kao

```
enum class Smjer {Nalijevo, Nadesno};
```

   Parametar "`orijentacija`" predstavlja pravac u kojem robot trenutno gleda. Tip ovog parametra je "`Pravci`", što je pobrojani tip koji je (također na globalnom nivou) deklariran kao

```
enum class Pravci {Sjever, Sjeveroistok, Istok, Jugoistok, Jug, Jugozapad, Zapad,
  Sjeverozapad};
```

   Značenja pojedinih pobrojanih konstanti u ovom tipu su očigledna. Funkcija "`Rotiraj`" utiče na vrijednost parametra "`orijentacija`", tako da će on po završetku funkcije sadržavati novu orijentaciju robota nakon obavljene rotacije. U slučaju da je robot deaktiviran ova funkcija ne radi ništa (isto vrijedi za sve ostale funkcije, ukoliko nije rečeno drugačije).

   Funkcija "`Idi`" pomjera robota za broj koraka zadan parametrom "`korak`" u smjeru koji je određen parametrom "`orijentacija`", pri čemu se 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.

   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 "aktivan" ili "neaktivan", 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 "PrijaviGresku" ima parametar "kod_greske" tipa "KodoviGresaka" koji predstavlja pobrojani tip definiran kao

```
enum class KodoviGresaka {PogresnaKomanda, NedostajeParametar, SuvisanParametar,
  NeispravanParametar};
```

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!

Funkcija "IzvrsiKomandu" ima parametar "komanda" tipa "Komande" koji predstavlja pobrojani tip definiran kao

```
enum class Komande {Aktiviraj, Deaktiviraj, Nalijevo, Nadesno, Idi, Kraj};
```

Ova funkcija, u zavisnosti od vrijednosti parametra "komanda", 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", i on tada predstavlja vrijednost koja se prosljeđuje istoimenoj funkciji (tj. funkciji "Idi"). U svim ostalim slučajevima, vrijednost parametra "parametar" se ignorira. Preostali parametri funkcije "IzvrsiKomandu" predstavljaju informaciju o poziciji i orijentaciji robota.

Komunikacija između korisnika i robota vrši se posredstvom funkcije "UnosKomande" koja ima sljedeći prototip:

```
bool UnosKomande(Komande &komanda, int &parametar, KodoviGresaka &kod_greske);
```

Ova funkcija očekuje od korisnika da zada unos komande putem tastature. Legalne komande su sljedeće (u komandama "A+" i "A-" znakovi "+" i "-" su dio komande a ne parametar):

Komanda:	Značenje
A+	Aktivira robota
A–	Deaktivira robota
L	Rotira robota nalijevo
D	Rotira robota nadesno
I *korak*	Pomjera robota za navedeni broj koraka
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. Komanda "I" je praćena cijelim brojem koji predstavlja broj koraka koji će robot napraviti. 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).

Na početku rada, robot je aktivan, nalazi se na poziciji (0, 0) i gleda na sjever. Napisane funkcije treba demonstrirati u glavnom programu u kojem će se u petlji zahtijevati unos komande pozivom funkcije "UnosKomande", nakon čega će se odgovarajuća komanda izvršiti (pozivom

funkcije "`IzvrsiKomandu`") ili će se prijaviti 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:

```
Robot je aktivan, nalazi se na poziciji (0,0) i gleda na sjever.
Unesi komandu: D
Robot je aktivan, nalazi se na poziciji (0,0) i gleda na sjeveroistok.
Unesi komandu: D
Robot je aktivan, nalazi se na poziciji (0,0) i gleda na istok.
Unesi komandu: I
Komanda trazi parametar koji nije naveden!
Unesi komandu: I5
Robot je aktivan, nalazi se na poziciji (5,0) i gleda na istok.
Unesi komandu: A-
Robot je neaktivan, nalazi se na poziciji (5,0) i gleda na istok.
Unesi komandu: I3
Robot je neaktivan, nalazi se na poziciji (5,0) i gleda na istok.
Unesi komandu:     D
Robot je neaktivan, nalazi se na poziciji (5,0) i gleda na istok.
Unesi komandu: A+
Robot je aktivan, nalazi se na poziciji (5,0) i gleda na istok.
Unesi komandu: D
Robot je aktivan, nalazi se na poziciji (5,0) i gleda na jugoistok.
Unesi komandu: I 4
Robot je aktivan, nalazi se na poziciji (9,-4) i gleda na jugoistok.
Unesi komandu: D
Robot je aktivan, nalazi se na poziciji (9,-4) i gleda na jug.
Unesi komandu: S
Nerazumljiva komanda!
Unesi komandu: IXY2
Parametar komande nije ispravan!
Unesi komandu: D
Robot je aktivan, nalazi se na poziciji (9,-4) i gleda na jugozapad.
Unesi komandu: D
Robot je aktivan, nalazi se na poziciji (9,-4) i gleda na zapad.
Unesi komandu: I2XY
Parametar komande nije ispravan!
Unesi komandu: I0
Robot je aktivan, nalazi se na poziciji (9,-4) i gleda na zapad.
Unesi komandu: I-1
Robot je aktivan, nalazi se na poziciji (10,-4) i gleda na zapad.
Unesi komandu: I3
Robot je aktivan, nalazi se na poziciji (7,-4) i gleda na zapad.
Unesi komandu: L2
Zadan je suvisan parametar nakon komande!
Unesi komandu: L
Robot je aktivan, nalazi se na poziciji (3,-4) i gleda na jugozapad.
Unesi komandu: KK
Zadan je suvisan parametar nakon komande!
Unesi komandu: K
Dovidjenja!
```

U slučaju da se prilikom kretanja robota premaši opseg tipa "`int`", odgovarajuća koordinata robota koja bi izašla izvan opsega treba da se postavi tačno na granicu opsega, kao da se oko polja po kojem se kreće robot nalazi "zid" koji ne da robotu da ide dalje iza zida. Na primjer, ukoliko pretpostavimo da je tekuća x koordinata robota 9990, a da je maksimalna vrijednost koja može stati u tip "`int`" 10000 (nije tolika, ali to je samo primjer), nakon pokušaja kretanja na istok u iznosu od recimo 50 koraka (uglavnom, više od 10), nova x koordinata robota treba biti 10000. Pri tome, ne smijete unaprijed pretpostaviti koliki je maksimalni opseg za tip "`int`" (npr. osloniti se na to da je na kompajleru koji koristimo taj opseg od -2147483648 do 2147483647), nego te informacije trebate saznati programskim putem.

VAŽNA NAPOMENA: 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!

2. Godine 1742. njemački matematičar Christian Goldbach je, eksperimentišući sa brojevima, naslutio da se svaki parni broj veći od 2 može napisati kao zbir dva prosta broja i to često na više različitih načina (recimo, imamo $24 = 5 + 19 = 7 + 17 = 11 + 13$). Mada je Goldbach ovu pretpostavku provjerio na velikom broju primjera, on nije uspio dokazati da ona zaista uvijek vrijedi, te je uputio pismo Leonhardu Euleru sa molbom da proba dokazati ili opovrgnuti tu hipotezu. Euleru to nije pošlo za rukom, ali interesantno je da do današnjeg dana (dakle, skoro 300 godina) niko nije uspio u tome (bez obzira na brojne pokušaje), tako da je ova pretpostavka, poznata kao Goldbachova hipoteza, do danas ostala kao jedna od najpoznatijih nedokazanih hipoteza u teoriji brojeva. Ova hipoteza je do danas testirana grubom silom za sve brojeve manje od $4 \cdot 10^{18}$ i pritom nije ni jednom zakazala, tako da se vjeruje da je ona tačna (mada do danas nije pružen dokaz da ona zaista vrijedi za apsolutno sve parne brojeve veće od 2).

   Vaš zadatak je da napravite funkciju "`Goldbach`" koja ima tri cjelobrojna parametra "`n`", "`p`" i "`q`" (tipa "`int`"). Funkcija za zadani prirodni broj n, koji predstavlja prvi parametar, treba da nađe dva prosta broja p i q takva da je $n = p + q$ (ukoliko oni postoje), i da nađene vrijednosti p i q smjesti redom u parametre "`p`" i "`q`". U slučaju da se takva rastava ne pronađe, funkcija treba baciti izuzetak tipa "`logic_error`" uz prateći tekst "Rastava ne postoji" (ovo će se jedino desiti za $n \leq 2$ ili za neke neparne vrijednosti n, s obzirom da je Goldbachova hipoteza sigurno tačna za sve parne brojeve veće od 2 koji mogu stati u tip "`int`" u jeziku C++). Treba obratiti pažnju da ovakva rastava može postojati i za neke neparne vrijednosti n (npr. $15 = 2 + 13$). U slučaju da postoji više rastava oblika $n = p + q$, treba uzeti takvu rastavu kod koje p ima najmanju a q najveću moguću vrijednost (tako za $n = 24$ treba uzeti $p = 5$ i $q = 19$, a ne recimo $p = 7$ i $q = 17$). Napisanu funkciju testirajte u testnom programu koji za broj unesen sa tastature ispisuje rastavu na dva prosta sabirka ili informaciju da takva rastava ne postoji. Dijalozi između korisnika i programa trebaju izgledati kao na sljedećim slikama:

```
Unesi broj: 24
24 je zbir prostih brojeva 5 i 19
```

```
Unesi broj: 15
15 je zbir prostih brojeva 2 i 13
```

```
Unesi broj: 37
37 nije zbir dva prosta broja!
```

3. Napišite generičku funkciju "`ZbirKontejnera`" koja ima dva parametra koji su neki kontejnerski tipovi podataka, ne nužno iste vrste i ne nužno istog tipa elemenata (recimo, jedan može biti dek cijelih brojeva, a drugi moderni niz realnih brojeva). Pri tome se za ove kontejnere *ne pretpostavlja* da se mogu indeksirati, odnosno operator "`[]`" ne mora biti za njih podržan (na Predavanju 7_b ćete vidjeti da ima i takvih kontejnera, recimo tip "`list`"). Jedino što se o njima zna da podržavaju funkcije "`begin`" i "`end`" koje vraćaju kao rezultat odgovarajuće iteratore, koji ne moraju nužno podržavati ništa više od onoga što svaki iterator mora da podržava (nisu svi iteratori jednako moćni s aspekta operacija koje podržavaju), a to su operacije dereferenciranja("`*`"), dodjele ("`=`"), inkrementiranja ("`++`"), te poređenja na jednakost i različitost ("`==`" i "`!=`"). Čak ni funkcija "`size`" ne mora biti podržana za kontejner (primjer kontejnerskog tipa koji ne podržava ni funkciju "`size`" je tip "`forward_list`"). Kao rezultat, funkcija treba da vrati vektor koji se sastoji od zbirova odgovarajućih elemenata prvog i drugog kontejnera (podrazumijeva se da se elementi mogu sabirati). Recimo, ukoliko je prvi parametar dek cijelih brojeva čiji su elementi 3, 5, 2 i 8, a drugi parametar moderni niz realnih brojeva čiji su elementi 2.5. 1.72, -1 i 4.9, funkcija kao rezultat treba da vrati vektor čiji su elementi 5.5, 6.72, 1 i 12.9 (tip elemenata vektora treba biti onakav kakav je tip rezultata sabiranja elemenata prvog i drugog kontejnera). U slučaju da je jedan kontejner kraći od drugog, treba smatrati da nedostajući elementi imaju vrijednosti koje su podrazumijevane vrijednosti za tip elemenata tog kontejnera (npr. 0 za brojčane tipove, prazan string za stringove, itd.).

   Napisanu funkciju demonstrirajte u kratkom testnom programu, koji traži unos dužina i elemenata dvije sekvence, koje treba smjestiti u vektor realnih brojeva, a koji zatim računa i ispisuje na ekran njihov zbir. Dijalog između korisnika i programa treba da izgleda kao na sljedećoj slici:

```
Duzina prvog kontejnera: 5
Elementi prvog kontejnera: 5 2 8 3 6
Duzina drugog kontejnera: 3
Elementi druge sekvence: 6 1 5
Zbir kontejnera: 11 3 13 3 6
```

NAPOMENA: Bez obzira što testni program radi samo s vektorima realnih brojeva, napisana funkcija će se testirati s više različitih kontejnera, različitih tipova elemenata. Ukoliko ne znate napraviti dovoljno općenitu funkciju koliko se traži u zadatku, napravite je da radi za što više specijalnih slučajeva. Dobićete nešto bodova i na takvu izvedbu.

4. Napravite generičku funkciju "`SortirajPoSumiRedova`" 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 sabirati i porediti po veličini (tj. da je za njih podržana operacija sabiranja, te operacije poređenja po veličini), a koja transformira tu matricu tako da ona postane sortirana po sumi redova u opadajući poredak, u smislu da nakon sortiranja njen prvi red bude onaj red koji ima najveću sumu elemenata, i tako dalje sve do posljednjeg reda, koji je red s najmanjom sumom elemenata. Ukoliko dva reda imaju istu sumu elemenata, tada prije treba da dođe onaj red koji dolazi ranije u leksikografskom poretku (podsjetimo se da se leksikografski poredak određuje na osnovu prvog para različitih elemenata, na primjer vektor čiji su elementi 3, 5, 2, 7, 1 i 9 dolazi leksikografski prije vektora čiji su elementi 3, 5, 8, 4 i 6, jer je $2 < 8$). 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 "`algorithm`", kojoj treba proslijediti odgovarajuću funkciju kriterija, koju treba izvesti kao imenovanu funkciju nazvanu "`Kriterij`" (šta će biti parametri ove funkcije, zaključite sami). Pri tome će ova funkcija kriterija također morati biti generička funkcija, s obzirom da tip elemenata matrice nije unaprijed poznat. Vodite računa da ćete, kada budete slali generičku funkciju kao parametar funkciji "`sort`", morati koristiti eksplicitnu specifikaciju parametra šablona, jer je dedukcija tipa moguća samo na osnovu parametara koji se prosljeđuju funkciji, a kad funkciju kriterija prosljeđujemo kao parametar drugoj funkciji, tom prilikom ne zadajemo joj nikakve parametre.

   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. Dijalog između korisnika i programa treba da izgleda poput sljedećeg:

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

Matrica nakon sortiranja:
9 8 9
4 5 2 1 1 3 2
2 5 1 3 4
3 3 2 3
```

5. Neka su date dvije sekvence $a_1, a_2, \ldots, a_m$ i $b_1, b_2, \ldots, b_n$, kao i funkcija f s dva argumenta. Matrica $\boldsymbol{C}$ formata $m \times n$ čiji se elementi $c_{i,j}$ računaju po formuli $c_{i,j} = f(a_i, b_j)$ za sve vrijednosti indeksa $i = 1, 2, \ldots, m$ i $j = 1, 2, \ldots, n$ naziva se *generalizirani spoljašni proizvod* (engl. *generalized outer product*) ove dvije sekvence pomoću funkcije f. Vaš zadatak je da napravite generičku funkciju nazvanu "`GeneraliziraniSpoljasnjiProizvod`". Prva dva parametra funkcije su pokazivači ili iteratori koji omeđuju sekvencu $a_1, a_2, \ldots, a_m$, a naredna dva pokazivači ili iteratori koji omeđuju sekvencu $b_1, b_2, \ldots, b_n$. Tipovi prva dva parametra moraju biti identični, kao i tipovi naredna dva parametra, ali tipovi prva dva parametra ne moraju biti identični tipovima naredna dva parametra (npr. prva dva parametra mogu biti pokazivači na objekte tipa "`double`", a naredna dva parametra iteratori za dek cijelih brojeva). To, između ostalog, povlači i da elementi ove dvije sekvence ne

moraju nužno biti istog tipa. Što se tiče funkcije f, ona može biti bilo imenovana funkcija, bilo lambda funkcija, bilo lambda zatvorenje (o kojima se govori na Predavanju 7_b). Jedino ograničenje je da ona mora moći primiti elemente ove dvije sekvence kao parametre. Tip vrijednosti koju vraća funkcija f može biti bilo kakav, i tog će tipa biti elementi matrice $\boldsymbol{C}$.

Ono što funkcija "`GeneraliziraniSpoljasnjiProizvod`" treba da uradi je da dinamički alocira matricu $\boldsymbol{C}$ postupkom *fragmentirane alokacije* i da popuni tako kreiranu matricu odgovarajućim vrijednostima tako da ona zaista bude generalizirani spoljašni proizvod zadane dvije sekvence. 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.

Napisanu funkciju demonstrirajte u kratkom testnom programu, koji traži unos dužina i elemenata dvije sekvence, od kojih prvu treba smjestiti u vektor a drugu u dek realnih brojeva, a koji zatim računa i ispisuje na ekran elemente njihovog generaliziranog spoljašnjeg proizvoda pomoću funkcije $f(x, y) = x + 2\,y$, te na kraju oslobađa svu alociranu memoriju. Dijalog između korisnika i programa treba da izgleda poput sljedećeg:

```
Duzina prve sekvence: 3
Elementi prve sekvence: 5 2 8
Duzina druge sekvence: 4
Elementi druge sekvence: 1 3 6 2
Generalizirani spoljasnji proizvod uz f(x,y)=x+2y:
7 11 17 9
4 8 14 6
10 14 20 12
```

NAPOMENA: 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.