Zadaća 2.

Ova zadaća nosi ukupno 4 poena. Prvi zadatak nosi 1 poen, drugi zadatak nosi 0,6 poena, dok posljednja dva zadatka nose po 1,2 poena. Svi zadaci se mogu uraditi na osnovu gradiva sa prvih šest predavanja i pretpostavljenog predznanja iz predmeta "Osnove računarstva". Rok za predaju ove zadaće je ponedjeljak, 4. IV 2016. do 20.00.

NAPOMENA 1: U slučaju da Vam zbog bilo kojeg razloga zatreba maksimalna vrijednost koja može da stane u neki tip podataka "*tip*", legalan način da se dobije ta vrijednost, neovisno od upotrijebljenog kompajlera, operativnog sistema i računarske arhitekture na kojoj se program izvršava, je upotrebom sljedeće konstrukcije:

```
std::numeric_limits<tip>::max()
```

Ova konstrukcija zahtijeva uključenje biblioteke "`<limits>`" u program.

NAPOMENA 2: Ukoliko prilikom autotestiranja u bilo kojem autotestu bude izazvano nešto što se smatra nedefiniranim ponašanjem, taj autotest neće biti priznat. Za studente koji to ne znaju, u nedefinirano ponašanje spada i pojava da u slučaju prekoračenja očekivani pozitivni rezultat postaje negativan. Zaista, to se dešava samo na računarskim arhitekturama koje negativne brojeve čuvaju u zapisu tzv. drugog komplementa (bez obzira što su takvi svi PC računari, to je ipak nedefinirano ponašanje, te se validni programi ne bi trebali oslanjati na to).

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 90° 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, Istok, Jug, Zapad};
```

   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`". 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 trenutno` *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, određena istoimenim parametrima, dok "*orijentacija*" može biti "sjever", "jug", "istok" ili "zapad", ovisno 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 bilo šta što ne predstavlja ispravan cijeli broj (uključujući suvišne znakove iza broja) tretiraju se 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. 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 trenutno aktivan, nalazi se na poziciji (0,0) i gleda na sjever.
Unesi komandu: D
Robot je trenutno 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 trenutno aktivan, nalazi se na poziciji (5,0) i gleda na istok.
Unesi komandu: A-
Robot je trenutno neaktivan, nalazi se na poziciji (5,0) i gleda na istok.
Unesi komandu: I3
Robot je trenutno neaktivan, nalazi se na poziciji (5,0) i gleda na istok.
Unesi komandu: D
Robot je trenutno neaktivan, nalazi se na poziciji (5,0) i gleda na istok.
Unesi komandu: A+
Robot je trenutno aktivan, nalazi se na poziciji (5,0) i gleda na istok.
Unesi komandu: D
Robot je trenutno aktivan, nalazi se na poziciji (5,0) i gleda na jug.
Unesi komandu: I4
Robot je trenutno aktivan, nalazi se na poziciji (5,-4) i gleda na jug.
Unesi komandu: D
Robot je trenutno aktivan, nalazi se na poziciji (5,-4) i gleda na zapad.
Unesi komandu: S
Nerazumljiva komanda!
Unesi komandu: IXY2
Parametar komande nije ispravan!
Unesi komandu: I2XY
Parametar komande nije ispravan!
Unesi komandu: I0
Robot je trenutno aktivan, nalazi se na poziciji (5,-4) i gleda na zapad.
Unesi komandu: I-1
Robot je trenutno aktivan, nalazi se na poziciji (6,-4) i gleda na zapad.
Unesi komandu: I3
Robot je trenutno aktivan, nalazi se na poziciji (3,-4) i gleda na zapad.
Unesi komandu: L2
Zadan je suvisan parametar nakon komande!
Unesi komandu: L
Robot je trenutno aktivan, nalazi se na poziciji (3,-4) i gleda na jug.
Unesi komandu: KK
Zadan je suvisan parametar nakon komande!
Unesi komandu: K
Dovidjenja!
```

Prilikom testiranja zadatka pretpostavljaće se da će pozicija robota uvijek moći biti predstavljena u tipu "`int`", odnosno *neće se testirati situacije u kojima dolazi do izlaska izvan opsega tipa* "`int`".

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. Nepoštovanje ovog pravila biće kažnjeno davanjem 0 poena na čitav zadatak!

2. Napišite generičku funkciju "`Izdvajanje`" sa tri parametra. Prva dva parametra (nazovimo ih "`p1`" i "`p2`") ograničavaju blok elemenata proizvoljnog cjelobrojnog tipa pri čemu "`p1`" pokazuje na prvi elemenat toga bloka a "`p2`" tačno iza kraja. Oba parametra mogu biti bilo pokazivači bilo iteratori, ali su oba istog tipa. Funkcija treba da kao rezultat vrati *pokazivač* na dinamički alocirani niz elemenata koji će se sastojati od svih savršenih brojeva pronađenih u proslijeđenom bloku, bez duplikata, u onom redoslijedu kako se javljaju u bloku. Pri tome, savršeni brojevi su oni brojevi koji su jednaki sumi svih svojih djelilaca ne računajući njega samog. Primjeri savršenih brojeva su brojevi 6 $(1 + 2 + 3 = 6)$ i 28 $(1 + 2 + 4 + 7 + 14 = 28)$. Ukoliko u proslijeđenom bloku elemenata nema savršenih brojeva funkcija treba da vrati nul-pokazivač. Funkcija smješta broj nađenih savršenih brojeva (koji je tipa "`int`") u svoj treći parametar.

Pored toga, funkcija treba da modificira proslijeđeni blok elemenata tako što će iz njega izbaciti sve elemente čiji je broj djelilaca jednak broju djelilaca ma kojeg drugog elementa u bloku. Na primjer, ukoliko se u proslijeđenom bloku elemenata nalaze brojevi 4 i 9, treba ih oba izbaciti, jer imaju isti broj djelilaca (to su brojevi 1, 2 i 4 za prvi broj te 1, 3 i 9 za drugi broj). Također, potrebno je sortirati tako modificirani blok u rastući poredak po sumi pozitivnih djelilaca, koristeći funkciju "`sort`" iz biblioteke "`algorithm`" uz pogodno definiranu funkciju kriterija koja se prosljeđuje kao lambda funkcija. Pri tome, ukoliko dva broja imaju istu sumu pozitivnih djelilaca, veći broj treba da dođe prije manjeg. Pošto može doći do promjene veličine bloka elemenata, parametar "`p2`" treba biti referenca i funkcija ga treba ažurirati ukoliko dođe do smanjenja veličine bloka. U slučaju nedostatka memorije funkcija treba da baci izuzetak tipa "`domain_error`" uz prateći tekst "Nedovoljno memorije!"

Kao primjer, za ulazni blok elemenata 28, 6, 10, 2, 48, 213, 5 i 54, funkcija u dinamički alocirani niz treba da smjesti brojeve 28 i 6, dok izvorni blok nakon poziva funkcije treba da se sastoji od elemenata 28, 54 i 48. Zaista, 28 i 6 su jedini savršeni blokovi među navedenim brojevima. Dalje, broj djelilaca elemenata ovog bloka respektivno glasi 6, 4, 4, 2, 10, 4, 2 i 8, tako da nakon odstranjenja ostaju elementi 28, 48 i 54. Njihova suma djelilaca respektivno glasi 56, 124 i 120, što nakon sortiranja daje 28, 54 i 48.

Obavezno napišite i "`main`" funkciju u kojoj se sa tastature unose brojevi u proizvoljan kontejner sve dok se ne unese broj -1 koji se ne razmatra. Potrebno je ispisati modificirani kontejner te sve savršene brojeve odvojene znakom zarez pri čemu iza posljednjeg broja nema zareza. Dijalog između korisnika i programa treba da izgleda poput sljedećeg:

```
Unesite brojeve: 28 6 10 2 48 213 5 54 -1
Modificirani kontejner: 28, 54, 48
Savrseni brojevi: 28, 6
```

U slučaju da funkcija "`Izdvajanje`" baci izuzetak potrebno je samo ispisati njegov prateći tekst.

VAŽNA NAPOMENA: U zadatku je neophodno koristiti pokazivačku/iteratorsku aritmetiku, odnosno zabranjeno je korištenje indeksiranja poput "`p[i]`" i trivijalna simulacija indeksiranja tj. korištenje izraza poput "`*(p + i)`". Također, mada je funkcija "`Izdvajanje`" generička funkcija, pretpostavlja se da će joj biti prosljeđivani samo blokovi čiji elementi zadovoljavaju koncept cjelobrojnosti, odnosno koji su nekog cjelobrojnog tipa (ali ne nužno tipa "`int`"). Konačno, strogo je zabranjeno kreiranje ikakvih pomoćnih kontejnera (nizova, vektora, dekova itd.) u ovoj funkciji (nepoštovanje ovog pravila biće kažnjeno davanjem 0 poena na čitav zadatak)!

3. Napišite generičku funkciju "`Izmijeni3DKontejner`" koja kao parametre redom prihvata:
   - Referencu na trodimenzionalni (3D) kontejner sa direktnim pristupom (tj. čijim se elementima može pristupati putem indeksa), a čiji su elementi proizvoljnog tipa. Pri tome, 3D kontejener je kontejner čiji su elementi 2D kontejneri, tako da to može biti recimo vektor vektôrā dekôvā, ili neki 3D niz, ili nešto treće.
   - Funkciju "`Fun`" koja kao parametre prihvata dva broja "`m`" i "`n`" standardnog cjelobrojnog tipa (tj. tipa "`int`"), a kao rezultat vraća vrijednost istog tipa kakav je i tip elemenata u kontejneru koji je prvi parametar.
   - Dva cijela broja "`p`" i "`q`" (tipa "`int`") koji imaju podrazumijevanu vrijednost 0.

   Funkcija treba da iz proslijeđenog 3D kontejnera zamijeni svako pojavljivanje broja kojeg vraća funkcija "`Fun`" kad joj se proslijede vrijednosti "`p`" i "`q`" sa njegovim kvadratom ukoliko je to moguće (tj. ukoliko je kvadrat dovoljno mali da može stati u element odgovarajućeg tipa), u suprotnom ne treba modificirati pronađeni broj. Kao rezultat, funkcija treba da vrati pronađeni broj prije modifikacije (tj. broj koji je vratila funkcija "`Fun`"). U slučaju da broj kojeg je vratila funkcija "`Fun`" nije pronađen nigdje u kontejneru, treba baciti izuzetak tipa "`range_error`" sa pratećim tekstom "Nije pronadjen broj!".

   Za potrebe testiranja ove funkcije, potrebno je da napravite još dvije pomoćne funkcije nazvane "`PascalovTrougao`" i "`NewtonovBinomniKoeficijent`". Obje funkcije primaju dva parametra "`m`" i "`n`" tipa "`int`" i vraćaju rezultat koji je također tipa "`int`". Prva funkcija ("`PascalovTrougao`") vraća kao rezultat broj koji se nalazi na n-toj poziciji u m-tom redu Pascalovog trougla (brojanje redova i elemenata unutar reda počinje od jedinice), pri čemu račun treba obaviti po definiciji

Pascalovog trougla (i-ti red ima i elemenata, prvi i posljednji element u svakom redu je 1, a svaki ostali element jednak je zbiru elemenata koji se nalaze tačno iznad njega i njegovog susjeda sa lijeve strane). Druga funkcija "`NewtonovBinomniKoeficijent`" vraća kao rezultat binomni koeficijent "n nad k", koji je definiran formulom

$$\binom{n}{k} = \frac{n!}{k!\,(n-k)!}$$

Pri tome, računanje treba obaviti tako da rezultat bude ispravan kad god on može stati u tip "`int`". Recimo, za $n = 1000$ i $m = 998$ funkcija treba vratiti ispravan rezultat 499500, bez obzira na činjenicu da je 1000! broj koji ima 2568 cifara. U slučaju da se ovim funkcijama pošalju besmisleni parametri, one treba da bace izuzetak tipa "`domain_error`" uz prateći tekst "Neispravni parametri!".

Napisane funkcije iskoristite u testnom programu ("`main`" funkciji) u kojem se demonstrira rad funkcije "`Izmijeni3DKontejner`" na način da joj se proslijedi 3D kontejner organiziran kao vektor vektôrā vektôrā, čiji se elementi unose sa tastature (dimenzije kontejnera se također prethodno unose sa tastature), zatim funkcija "`PascalovTrougao`", te brojevi "`p`" i "`q`" koji se unose sa tastature. Dijalog između korisnika i programa treba da izgleda poput sljedećeg (elementi kontejnera se ispisuju "sprat po sprat", pri čemu je prvi indeks "sprat", a drugi i treći su red odnosno kolona unutar tekućeg sprata; elementi se ispisuju na 5 mjesta širine poravnato ulijevo):

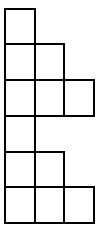
```
Unesite dimenzije 3D kontejnera: 3 3 3
Unesite elemente: 1 2 3 4 5 6 7 8 9 9 8 7 6 5 4 3 2 1 1 2 3 4 5 6 7 8 9
Unesite brojeve p i q: 6 2
Kontejner nakon zamjene broja 5 brojem 25 glasi:
1    2    3
4    25   6
7    8    9

9    8    7
6    25   4
3    2    1

1    2    3
4    25   6
7    8    9
```

NAPOMENA: Da bi probem uopće imao smisla, elementi kontejnera moraju biti nekog tipa za koji su podržane one operacije koje su neophodne da se tijelo funkcije izvede kako treba. Funkciji se neće slati kontejneri koji sadrže nešto za šta to ne vrijedi (npr. neće se slati kontejneri sa elementima tipa "`string`" za koje je kvadriranje besmisleno).

4. Napravite funkcije "`AlocirajFragmentirano`" i "`AlocirajKontinualno`" koje kao parametre primaju referencu na konstantni vektor čiji su elementi nekog cjelobrojnog tipa (ali nije tačno određeno kojeg), te cjelobrojnu promjenljivu (tipa "`int`") nazvanu "`koliko_puta`". Obje funkcije treba da rade istu stvar, a to je da dinamički alociraju grbavu matricu čiji su elementi tipa "`int`" (prva koristeći fragmentiranu, a druga kontinualnu alokaciju), pri čemu je dužina i-tog reda određena i-tim elementom vektora koji je zadan kao parametar. Na primjer, ukoliko je nekoj od tih funkcija proslijeđen vektor čiji su elementi 8, 5, 4, 1, 3, 4, i 5, tada prvi red alocirane grbave matrice treba imati 8 elemenata, drugi red 5 elemenata, itd. Pored toga, sve kreirane redove treba umnožiti onoliko puta koliko iznosi vrijednost parametra "`koliko_puta`" (odnosno, kao da se čitav sadržaj zadanog vektora ponavlja "`koliko_puta`" puta). Na primjer, ukoliko vektor ima elemente 1, 2 i 3, a parametar "`koliko_puta`" vrijednost 2, kreirana grbava matrica treba imati sljedeću strukturu:



U slučaju da bilo koji od elemenata vektora ima besmislenu vrijednost (negativnu ili nulu), funkcije trebaju baciti izuzetak tipa "`domain_error`", uz prateći tekst "Neispravan vektor!". Isti izuzetak, ali uz prateći tekst "Neispravan broj ponavljanja!" treba baciti u slučaju da je parametar "`koliko_puta`" besmislen (negativan ili nula). Vodite računa da ni u kom slučaju prilikom bacanja izuzetaka ne smije da dođe do curenja memorije, odnosno u slučaju da dođe do bilo kakvih problema (uključujući i nedostatak memorije), funkcije moraju da "počiste iza sebe" sve što su do tada alocirale prije nego što bace izuzetak.

Pored alokacije, opisane funkcije treba i da popune elemente kreiranih matrica. Pri tome, funkcija "`AlocirajKontinualno`" treba da svaki red popuni opadajućim slijedom uzastopnih prirodnih brojeva koji uvijek završava sa 1 (posljednji element svakog reda treba biti 1), dok funkcija "`AlocirajFragmentirano`" treba da svaki red popuni rastućim slijedom uzastopnih prirodnih brojeva koji uvijek završava sa brojem koji je jednak dužini najdužeg reda matrice. Na primjer, ukoliko vektor ima elemente 1, 2 i 3, a parametar "`koliko_puta`" ima vrijednost 3, funkcija "`AlocirajKontinualno`" treba da kreira matricu čiji su elementi popunjeni na sljedeći način:

1		
2	1	
3	2	1
1		
2	1	
3	2	1
1		
2	1	
3	2	1

S druge strane, funkcija "`AlocirajFragmentirano`" za ove iste parametre treba popuniti grbavu matricu na sljedeći način:

3		
2	3	
1	2	3
3		
2	3	
1	2	3
3		
2	3	
1	2	3

Obje ove funkcije kao rezultat vraćaju pokazivač pomoću kojeg se može pristupiti elementima ovako alociranih matrica.

Pored ove dvije funkcije treba napraviti i treću funkciju nazvanu "`KreirajPoUvrnutomPravilu`", koja će pozivati prethodne dvije funkcije. Prvi parametar je referenca na vektor čiji su elementi nekog cjelobrojnog tipa (mada nije precizno određeno kojeg), drugi parametar nazvan "`koliko_puta`" je standardnog cjelobrojnog tipa (tipa "**`int`**"), dok je treći parametar nazvan "`fragmentirano`" logičkog tipa i ima podrazumijevanu vrijednost "**`true`**". Ova funkcija prvo treba da provjeri da li je slijed brojeva pohranjen u vektoru periodičan (podsjetimo se da je neki slijed brojeva a_i periodičan ako i samo ako postoji takav broj T nazvan period takav da je $a_i = a_{i+\mathrm{T}}$ za sve vrijednosti i za koje su oba indeksa i i $i + \mathrm{T}$ smisleni). Ukoliko je odgovor potvrdan, vektor treba skratiti na dužinu njegovog osnovnog perioda (osnovni period je najmanji od svih mogućih perioda). Na primjer, ukoliko vektor sadrži slijed brojeva 1, 2, 3, 1, 3, 2 i 1, treba ga skratiti na dužinu 6 (jer je ovaj slijed periodičan sa osnovnim periodom 6), dok vektor koji sadrži slijed brojeva 1, 2, 3, 1 i 2 treba skratiti na dužinu 3 (jer je to osnovni period ovog slijeda brojeva). Međutim, ukoliko slijed brojeva u vektoru nije periodičan, treba taj slijed zamijeniti sa njegovim komplementom, koji se računa tako da se svi elementi slijeda oduzmu od najvećeg elementa slijeda uvećanog za 1. Na primjer, ukoliko vektor sadrži slijed 1, 4, 6, 8, 6, 5 i 4 (koji očito nije periodičan), ovaj slijed treba zamijeniti sa njegovim komplementom koji glasi 8, 5, 4, 1, 3, 4 i 5 (najveći element slijeda je 8, tako da se svi elementi oduzimaju od 9).

Nakon što se obavi tražena transformacija vektora, transformirani vektor treba proslijediti kao prvi parametar funkciji "AlocirajFragmentirano" ili "AlocirajKontinualno", ovisno o tome da li parametar "fragmentirano" ima vrijednost "**true**" ili "**false**". Parametar "koliko_puta" iz funkcije "KreirajPoUvrnutomPravilu" treba prosto proslijediti istoimenom parametru jedne od ove dvije funkcije (ovisno od toga koja se poziva). Na kraju, pokazivač koji vrate ove funkcije treba ujedno vratiti i kao rezultat iz funkcije "KreirajPoUvrnutomPravilu".

U slučaju da se bilo kojoj od tri funkcije koje se traže u zadatku kao parametar proslijedi prazan vektor, funkcije trebaju da bace tekstualni izuzetak koji se sastoji od teksta "Prazan vektor!".

Eventualne izuzetke bačene iz funkcija "AlocirajFragmentirano" ili "AlocirajKontinualno" ne treba hvatati unutar funkcije "KreirajPoUvrnutomPravilu", nego oni samo treba da "prođu kroz nju" tako da mogu eventualno biti obrađeni na nekom višem nivou obrade (recimu unutar "main" funkcije). Isto vrijedi i za sistemski generirane izuzetke (koji se recimo bacaju u slučaju neuspješne dinamičke alokacije).

Napisane funkcije demonstrirajte u "main" funkciji koja će za uneseni vektor v (prethodno se unosi broj njegovih elemenata) te unesene vrijednosti n i f (pri čemu se "1" tumači kao "**true**", a "0" kao "**false**"), pozvati funkciju "KreirajPoUvrnutomPravilu" prosljeđujući joj kao parametre v, n i f respektivno, a zatim preko pokazivača koji je vraćen kao povratna vrijednost ispisati sve elemente dinamički alocirane grbave matrice red po red, tako da svaki element zauzme prostor širine 3 karaktera, uz ravnanje ulijevo u predviđenom prostoru. U "main" funkciji je potrebno predvidjeti i hvatanje svih mogućih izuzetaka koji mogu biti bačeni. Slijede tri primjera kako treba izgledati dijalog između korisnika i programa:

```
Unesite broj elemenata vektora: 5
Unesite elemente vektora: 1 2 3 1 2
Unesite broj ponavljanja: 3
Odaberite alokaciju: 1 - fragmentirana, 0 - kontinualna: 1
Dinamicki alocirana matrica:
3
2  3
1  2  3
3
2  3
1  2  3
3
2  3
1  2  3

Unesite broj elemenata vektora: 4
Unesite elemente vektora: 1 2 -4 1
Unesite broj ponavljanja: 3
Odaberite alokaciju: 1 - fragmentirana, 0 - kontinualna: 1
Izuzetak: Neispravan vektor.

Unesite broj elemenata vektora: 3
Unesite elemente vektora: 1 2 4
Unesite broj ponavljanja: 2
Odaberite alokaciju: 1 - fragmentirana, 0 - kontinualna: 0
Dinamicki alocirana matrica:
4  3  2  1
3  2  1
1
4  3  2  1
3  2  1
1
```

VAŽNA NAPOMENA: Strogo je zabranjeno kreiranje ikakvih pomoćnih kontejnera (nizova, vektora, dekova itd.) unutar funkcija "AlocirajFragmentirano" ili "AlocirajKontinualno", osim onoga što ove funkcije zaista treba da kreiraju kao svoj zadatak. Nepoštovanje ovog pravila biće kažnjeno davanjem 0 poena na čitav zadatak!