Zadaci za Laboratorijsku vježbu 3

NAPOMENA: Studenti bi trebali da razmisle o zadacima koji će se raditi na laboratorijskoj vježbi prije nego što dođu na vježbu, tako da već u startu imaju osnovne ideje kako riješiti zadatke. U suprotnom, rad na vježbi neće biti produktivan. Zadatke koje studenti ne stignu uraditi za vrijeme vježbe, trebali bi ih samostalno uraditi kod kuće. Zadataka ovaj put ima malo više, ali treba imati u vidu da su zadaci 2, 3, 4 i 5 samo manje-više trivijalna prepravka zadatka 1.

1. Napišite funkciju "`IzdvojiElemente`" sa dva parametra, od kojih je prvi vektor cijelih brojeva (tipa "`int`"), a drugi logička vrijednost "tačno" ili "netačno". Funkcija treba da kao rezultat vrati novi vektor koji se sastoji od onih elemenata čija je suma cifara parna odnosno neparna, ovisno od toga da li drugi parametar ima vrijednost "tačno" ili "netačno". Elementi u novokreiranom vektoru trebaju biti u istom međusobnom poretku u kakvom su bili u izvornom vektoru. Za realizaciju funkcije obavezno koristite funkciju "`push_back`". Napisanu funkciju iskoristite u testnom programu koji traži da se prvo unese prirodan broj n, a nakon toga elementi vektora "`a`" koji ima n cjelobrojnih elemenata. Program zatim treba da kreira dva nova vektora "`b`" i "`c`", i da u vektor "`b`" prepiše sve brojeve iz vektora "`a`" sa parnom sumom cifara, a u vektor "`c`" sve brojeve iz vektora "`a`" sa neparnom sumom cifara. Konačno, program treba da u jednom redu ispiše brojeve sa parnom sumom cifara, a u drugom redu brojeve sa neparnim brojem cifara (moguće je da neki red ostane prazan, ukoliko nema niti jedan broj sa traženim svojstvom). Mogući dijalozi između programa i korisnika trebaju izgledati ovako:

```
Koliko zelite unijeti elemenata: 12
Unesite elemente: 15 23 18 -26 0 142 -333 73 11111 -312 9 5555555
15 -26 0 73 -312
23 18 142 -333 11111 9 5555555
```

```
Koliko zelite unijeti elemenata: 5
Unesite elemente: 24 73 112 -17 0
24 73 112 -17 0

```

```
Koliko zelite unijeti elemenata: 4
Unesite elemente: 32 -23 111 45

32 -23 111 45
```

   Za sve manipulacije sa vektorima koristite *isključivo rasponsku for-petlju*, osim za unos elemenata (u ovom trenutku još ne znate kako koristiti rasponsku for-petlju za unos, jer Vam za to trebaju reference). Možete pretpostaviti da će svi ulazni podaci biti korektni.

2. Prepravite prethodni program tako da se brojevi ispisuju razdvojeni *zarezom* a ne razmakom, pri čemu iza *posljednjeg broja u svakom redu ne treba da bude zarez*. Na primjer, za iste ulazne podatke kao u prethodnom zadatku, dijalozi između korisnika i programa trebali bi izgledati ovako:

```
Koliko zelite unijeti elemenata: 12
Unesite elemente: 15 23 18 -26 0 142 -333 73 11111 -312 9 5555555
15,-26,0,73,-312
23,18,142,-333,11111,9,5555555
```

```
Koliko zelite unijeti elemenata: 5
Unesite elemente: 24 73 112 -17 0
24,73,112,-17,0

```

```
Koliko zelite unijeti elemenata: 4
Unesite elemente: 32 -23 111 45

32,-23,111,45
```

   Ovaj put za ispis elemenata vektora *nemojte koristiti rasponsku for-petlju*, jer ćete u suprotnom teško postići da ne bude ispisan zarez iza posljednjeg elementa u svakom redu.

3. Prepravite prethodni program tako da umjesto tipa "`vector`" koristi tip "`deque`" (tj. da koristi dek umjesto vektora), i uvjerite se da sve i dalje radi potpuno isto.

4. U prethodnom programu, zamijenite poziv funkcije "`push_back`" unutar funkcije "`IzdvojiElemente`" (koju ste morali koristiti za njenu realizaciju) sa pozivom funkcije "`push_front`" i uporedite razliku.

5. U prethodnom programu, izvršite izmjene u funkciji "`IzdvojiElemente`" tako da se i dalje koristi funkcija "`push_front`", ali da rezultat koji daje funkcija bude isti kakav je bio u Zadatku 3. (tako da i ispis koji će program proizvesti bude isti kao u Zadatku 3.). Pri tome, u funkciji nije dozvoljeno kreiranje ikakvih pomoćnih kontejnera (nizova, vektora, dekova, itd.).

6. U linearnoj algebri, pod Kroneckerovim ili tenzorskim proizvodom $\boldsymbol{a} \otimes \boldsymbol{b}$ vektora $\boldsymbol{a}$ i $\boldsymbol{b}$ čiji su elementi a_i, $i = 1, 2, \ldots, m$ odnosno b_i, $i = 1, 2, \ldots, n$, podrazumijevamo matricu $\boldsymbol{C}$ formata $m \times n$ čiji se elementi računaju po formuli $c_{i,j} = a_i b_j$, $i = 1, 2, \ldots, m$, $j = 1, 2, \ldots, n$ (koristeći kompaktniju notaciju iz linearne algebre, ukoliko su $\boldsymbol{a}$ i $\boldsymbol{b}$ predstavljeni kao vektor kolone, vrijedi $\boldsymbol{a} \otimes \boldsymbol{b} = \boldsymbol{a}\,\boldsymbol{b}^{\mathrm{T}}$). Napravite funkciju "`KroneckerovProizvod`" koja prima kao parametre dva vektora cijelih brojeva (tipa "`int`"), a vraća kao rezultat matricu organiziranu kao vektor vektôrā koja predstavlja Kroneckerov proizvod vektora koji su zadani kao parametri (pri tome, funkcija se treba konzistentno ponašati čak i ukoliko je neki od ulaznih vektora prazan). Napisanu funkciju iskoristite u glavnom programu u kojem će se sa tastature unijeti elementi dva vektora, a zatim kreirati i prikazati na ekranu njihov Kroneckerov proizvod. Pri tome, ispis na ekranu treba biti takav da se za svaki element matrice zauzme prostor koji je za 1 veći nego što je širina najšireg broja u matrici, uz poravnavanje udesno (pod širinom broja podrazumijeva se broj njegovih cifara, eventualno uvećan za 1 ako je broj negativan, zbog prostora koji zauzima znak "−"). Za tu svrhu, u programu trebate napisati i još jednu funkciju nazvanu "`NajvecaSirina`" koja kao parametar prima matricu organiziranu kao vektor vektôrā cijelih brojeva, a vraća širinu najšireg broja u matrici. Funkcija treba ispravno da radi i ukoliko joj se ponudi grbava matrica, pa čak i prazna matrica (u posljednjem slučaju, funkcija treba vratiti 0 kao rezultat). Dijalog između korisnika i programa treba izgledati poput sljedećeg:

```
Unesite broj elemenata prvog vektora: 4
Unesite elemente prvog vektora: 3 124 -11 9
Unesite broj elemenata drugog vektora: 6
Unesite elemente drugog vektora: 42 1001 0 -213 16 5

   126   3003      0    -639     48     15
  5208 124124      0  -26412   1984    620
  -462 -11011      0    2343   -176    -55
   378   9009      0   -1917    144     45
```

Možete pretpostaviti da će svi ulazni podaci biti korektni. Za realizaciju ispisa matrice možete, ali i ne morate koristiti rasponsku *for*-petlju (najbolje je da probate obje varijante, sa i bez rasponske *for*-petlje).

7. Napišite funkciju "`PascalovTrougao`" (ili "`PascalovTrokut`", ovisno od Vaših jezičkih preferencija) koja kao parametar prima cijeli broj n. Ova funkcija treba kreirati strukturu "grbave matrice" sa n redova u kojoj prvi red ima jedan element, drugi red dva elementa, treći red tri elementa, itd. koju nakon kreiranja treba popuniti elementima Pascalovog trougla (trokuta) sa n redova. Funkcija treba da kao rezultat vrati upravo tako kreiranu "grbavu matricu". Funkciju testirajte u programu koji za unesenu vrijednost n sa tastature ispisuje Pascalov trougao (trokut) sa n redova. Prilikom ispisa, za svaki element treba zauzeti onoliko prostora koliko iznosi širina najšireg broja koji se javlja u trouglu (trokutu) uvećana za 1, uz ravnanje udesno (za određivanje ove širine, iskoristite istu funkciju "`NajvecaSirina`" koju ste pisali u prethodnom zadatku. Uglavnom, dijalog između programa i korisnika trebao bi izgledati poput sljedećeg;

```
Unesite broj redova: 7

  1
  1  1
  1  2  1
  1  3  3  1
  1  4  6  4  1
  1  5 10 10  5  1
  1  6 15 20 15  6  1
```

Pri tome, elemente Pascalovog trougla *nije dozvoljeno* računati preko binomnih koeficijenata, već treba koristiti činjenicu da je svaki element Pascalovog trougla jednak zbiru elemenata koji se nalaze tačno iznad njega, i njegovog susjeda sa lijeve strane.

Ukoliko se funkciji n kao parametar proslijedi 0, funkcija treba da vrati praznu matricu. Ukoliko joj se proslijedi negativan broj, funkcija treba da baci izuzetak tipa "`domain_error`" uz prateći tekst "Broj redova ne smije biti negativan". U glavnom programu ne trebate predvidjeti hvatanje ovog izuzetka, ali će biti testirano da li ga funkcija zaista baca.

8. Napišite funkciju "`IzdvojiRijec`" sa dva parametra, od kojih je prvi tipa "`string`", a drugi je cjelobrojan (tipa "`int`"). Prvi parametar predstavlja neku rečenicu, a drugi parametar redni broj riječi unutar te rečenice. Funkcija treba da izdvoji tu riječ iz rečenice, i da vrati kao rezultat tako izdvojenu riječ. Na primjer, ukoliko je kao prvi parametar zadan tekst " Na vrh brda vrba mrda" a kao drugi parametar broj 4, funkcija treba kao rezultat da vrati string "vrba". Ovdje pod pojmom riječ podrazumijevamo bilo koji slijed uzastopnih znakova koji nisu razmaci, a ispred kojeg se eventualno nalazi razmak (ili ništa), i iza kojeg eventualno slijedi razmak (ili ništa). Tako se, na primjer, u tekstu "Kiša pada.Trava raste" slijed znakova "pada.Trava" tretira kao *jedna riječ* (druga po redu), jer iza tačke *nema razmaka* (ovakav tretman vrijedi i u tekst procesorima poput Microsoft Word-a). Obratite pažnju da riječi mogu biti razdvojene sa više uzastopnih razmaka, kao i da na početku i kraju teksta može, ali i ne mora biti razmaka. Ukoliko je drugi parametar manji od 1 ili veći od broja riječi u rečenici, funkcija treba baciti izuzetak tipa "`range_error`" uz prateći indeks "Pogresan redni broj rijeci".

   Napisanu funkciju demonstrirajte u testnom programu u kojem se za prirodan broj n unesen sa tastature i rečenicu unesenu sa tastature ispisuje n-ta riječ te rečenice (pozivom napisane funkcije). U testnom programu obavezno predvidite hvatanje izuzetaka koji mogu biti bačeni iz funkcije. Dijalozi između korisnika i programa trebaju izgledati poput sljedećeg:

```
Unesite redni broj rijeci: 4
Unesite recenicu:   Na     vrh  brda   vrba  mrda
Rijec na poziciji 4 je vrba
```

```
Unesite redni broj rijeci: 6
Unesite recenicu:   Na     vrh  brda   vrba  mrda
IZUZETAK: Pogresan redni broj rijeci!
```

   Možete pretpostaviti da će na pitanje da se unese redni broj riječi zaista biti unesen broj.

Za potrebe realizacije zadataka 2., 3., 4. i 5. **dozvoljeno je** kopirati programski kôd zadatka 1.