Zadaća 1.

Ova zadaća nosi ukupno 4 poena, pri čemu svaki zadatak nosi po 1 poen. Svi zadaci se mogu uraditi na osnovu gradiva sa prva tri predavanja i pretpostavljenog predznanja iz predmeta "Osnove računarstva". Rok za predaju ove zadaće je subota, 1. IV 2017. do 22.00.

NAPOMENA: Tačan izgled dijaloga između korisnika i programa u testnim programima za zadatke nije precizno specificiran u postavci zadataka. Izgled dijaloga biće naknadno specificiran putem javnih autotestova, odnosno dijalog ćete prilagoditi tim autotestovima. Ostali detalji su precizno specificirani i ne ostavljaju nikakve nedoumice.

1. U teoriji brojeva, svi cijeli brojevi dijele se u dvije kategorije, ovisno od broja jedinica u njihovom binarnom zapisu. *Opaki brojevi* (engl. *evil numbers*) su brojevi koji imaju paran broj jedinica u binarnom zapisu, dok *odvratni brojevi* (engl. *odious numbers*) imaju neparan broj jedinica u binarnom zapisu (znak broja pri tome ne ne igra nikakvu ulogu).

   S obzirom da se u posljednje vrijeme u teorijskom računarstvu sve ozbiljnije razmatra ternarni brojni sistem (sa bazom 3), prirodno se javlja potreba da se ove definicije prilagode ternarnom brojnom sistemu. Stoga je predložena nova definicija, prema kojoj su opaki brojevi oni prirodni brojevi kod kojih se u ternarnom zapisu svaka cifra koja se uopće javlja u njima javlja paran broj puta, dok su odvratni brojevi oni kod kojih se u ternarnom zapisu svaka cifra koja se uopće javlja u njima javlja neparan broj puta. Na primjer, broj 56 je opak prema ovoj definiciji. Zaista, njegova ternarna reprezentacija je 2002 (s obzirom da je $56 = 2 \cdot 3^3 + 0 \cdot 3^3 + 0 \cdot 3^1 + 2 \cdot 3^0$), a obje cifre 0 i 2 koje se u toj reprezentaciji javljaju javljaju se paran broj puta. S druge strane, broj 67 je odvratan prema ovoj definiciji. Zaista, njegova ternarna reprezentacija je 2111 (s obzirom da je $67 = 2 \cdot 3^3 + 1 \cdot 3^3 + 1 \cdot 3^1 + 1 \cdot 3^0$), a cifre 1 i 2 koje se u toj reprezentaciji javljaju javljaju se neparan broj puta. Kako u ternarnoj logici uvijek imamo i treću mogućnost, sasvim je moguće da neki prirodan broj ne bude ni opak ni odvratan. Brojevi koji su bilo opaki, bilo odvratni prema ovoj definiciji, zajedničkim imenom ćemo nazvati *gadni brojevi*.

   Vaš zadatak je da napišete funkciju "`IzdvojiGadne`" koja prihvata dva parametra. Prvi parametar je vektor cijelih brojeva (tipa "**`int`**"), a drugi logička vrijednost "tačno" ili "netačno". Funkcija treba da kao rezultat vrati novi vektor koji se sastoji samo od onih brojeva iz vektora koji je zadan kao parametar koji su opaki ili koji su odvratni, ovisno od toga da li drugi parametar ima vrijednost "tačno" ili "netačno". Brojevi u vektoru koji se vraća kao rezultat trebaju biti u istom međusobnom poretku u kakvom su bili u izvornom vektoru. Pri tome, treba izostaviti suvišna pojavljivanja istog elementa. Recimo, ukoliko je izvorni vektor sadržavao broj 56 tri puta, a tražimo izdvajanje opakih brojeva, treba izdvojiti samo prvu pojavu ovog broja, tako da će se u rezultirajućem vektoru on pojaviti samo jedanput.

   Obavezno napišite i mali testni program ("`main`" funkciju) u kojoj ćete testirati napisanu funkciju na skupini brojeva koji se unose sa tastature.

2. Napišite funkciju "`OgledaloMatrica`" koja prihvata matricu realnih brojeva (realiziranu kao vektor vektôrā čiji su elementi tipa "**`double`**") a koja kao rezultat vraća matricu sastavljenu od "ogledala" matrice koja joj je proslijeđena kao parametar. Postoje horizontalna, vertikalna i kombinovana ogledala. Horizontalno ogledalo $h(\mathbf{M})$ matrice $\mathbf{M}$ se dobije kada obrnemo redoslijed kolona u matrici. Slično, vertikalno ogledalo $v(\boldsymbol{M})$ se dobije kada obrnemo redoslijed redova matrice $\mathbf{M}$. Na kraju, kombinovano ogledalo $hv(\mathbf{M})$ matrice $\mathbf{M}$ je kombinacija horizontalnog i vertikalnog ogledala, odnosno predstavlja matricu kojoj obrnemo i redoslijed kolona i redoslijed redova. Na primjer, neka imamo matricu $\mathbf{M}$ definiranu kao

$$\mathbf{M} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

   Njena ogledala redom glase:

$$h(\mathbf{M}) = \begin{pmatrix} 3 & 2 & 1 \\ 6 & 5 & 4 \end{pmatrix} \qquad v(\mathbf{M}) = \begin{pmatrix} 4 & 5 & 6 \\ 1 & 2 & 3 \end{pmatrix} \qquad hv(\mathbf{M}) = \begin{pmatrix} 6 & 5 & 4 \\ 3 & 2 & 1 \end{pmatrix}$$

Funkcija "`OgledaloMatrica`" treba da vrati matricu kod koje se u centru nalazi matrica **M**, proslijeđena kao parametar, sa lijeve i desne strane nalaze se njena horizontalna ogledala $h(\mathbf{M})$, sa gornje i donje strane nalaze se njena vertikalna ogledala $v(\mathbf{M})$, dok se u uglovima nalaze njena kombinovana ogledala $hv(\mathbf{M})$. Drugim riječima, treba vratiti matricu sljedećeg izgleda:

$$\begin{pmatrix} hv(\mathbf{M}) & v(\mathbf{M}) & hv(\mathbf{M}) \\ h(\mathbf{M}) & \mathbf{M} & h(\mathbf{M}) \\ hv(\mathbf{M}) & v(\mathbf{M}) & hv(\mathbf{M}) \end{pmatrix}$$

Na primjer, za konkretnu matricu **M** iz ranijeg primjera, vraćena matrica treba da bude sljedeća:

$$\begin{pmatrix} 6 & 5 & 4 & 4 & 5 & 6 & 6 & 5 & 4 \\ 3 & 2 & 1 & 1 & 2 & 3 & 3 & 2 & 1 \\ 3 & 2 & 1 & 1 & 2 & 3 & 3 & 2 & 1 \\ 6 & 5 & 4 & 4 & 5 & 6 & 6 & 5 & 4 \\ 6 & 5 & 4 & 4 & 5 & 6 & 6 & 5 & 4 \\ 3 & 2 & 1 & 1 & 2 & 3 & 3 & 2 & 1 \end{pmatrix}$$

Očigledno, ukoliko je ulazna matrica formata $m \times n$, izlazna će biti formata $3m \times 3n$. U skladu s tim, treba tretirati i rubne slučajeve. Konkretno, ukoliko se funkciji proslijedi posve prazna matrica (tj. "matrica" formata 0×0), rezultat treba biti ista takva matrica. Ukoliko se funkciji proslijedi "matrica" sa m praznih redova (tj. "matrica" formata $m \times 0$), izlazna "matrica" treba se sastojati od $3m$ praznih redova (tj. treba biti formata $3m \times 0$). Ukoliko ste funkciju "pametno" napisali, sve ovo će se desiti automatski "samo po sebi", bez potrebe da ove rubne slučajeve posebno tretirate ikakvim dodatnim uvjetima (što nije zabranjeno, ali bespotrebno komplicira stvari). Konačno, ukoliko se funkciji proslijedi "grbava matrica" (tj. vektor vektôrā u kojem nije ispunjeno da svi redovi imaju isti broj elemenata), treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Matrica nije korektna".

Napisanu funkciju demonstrirajte u testnom programu u kojem se sa tastature prvo traži unos dva cijela broja m i n, zatim elementi neke matrice formata $m \times n$. Program nakon toga treba da pozivom napisane funkcije kreira novu matricu (sa traženim ogledanjima) i da ispiše njene elemente na ekran, svaki red u posebnom redu na ekranu. Obavezno predvidite hvatanje izuzetaka koji bi eventualno mogli biti bačeni iz funkcije.

NAPOMENA: U konkretnom testnom programu koji se ovdje traži, izuzetak sa pratećim tekstom "Matrica nije korektna" nikad neće biti bačen, s obzirom da će se funkciji iz testnog programa uvijek prenositi parametar koji zaista ima strukturu matrice. Međutim, s obzirom da dobro napisana funkcija ne treba ništa da zna o tome u kakvom će okruženju biti korištena, funkcija mora da se zna "odbraniti" u slučaju da u nju ipak "uđu" neispravni podaci. Štaviše, za vrijeme autotestiranja, funkcije se zaista pozivati sa svakakvim ulaznim parametrima, a ne samo onima kakve ste vi predvidjeli u testnom programu.

3. Neka je dat neki niz brojeva a_1, a_2, a_3, …, a_n. Za neki njegov podniz a_p, a_{p+1}, a_{p+2}, …, a_q koji ima barem dva elementa (pri čemu je naravno $1 \leq p < q \leq n$) kažemo da je *monotono rastući* ukoliko je $a_p \leq a_{p+1} \leq a_{p+2} \leq \ldots \leq a_q$. Dalje, za podniz kažemo da je *maksimalni monotono rastući podniz* ukoliko je on monotono rastući, ali pri tome on sam nije podniz niti jednog drugog monotono rastućeg podniza (osim naravno sebe samog). Recimo, u nizu brojeva 1, 3, 2, 6, 7, 9, 10, 4, 3, 1, 7, podniz koji se sastoji od brojeva 2, 6, 7, 9 svakako jeste monotono rastući podniz, ali on nije maksimalni monotono rastući podniz, jer je on podniz monotono rastućeg podniza 2, 6, 7, 9, 10. Anlogno se definiraju monotono opadajući i maksimalni monotono opadajući podnizovi.

   Vaš zadatak je da napravite funkcije "`RastuciPodnizovi`" i "`OpadajuciPodnizovi`", od kojih obje primaju jedan parametar koji je vektor realnih brojeva, a vraćaju kao rezultat vektor vektôrā realnih brojeva. Elementi tog vektora vektôrā su vektori koji predstavljaju maksimalne monotono rastuće odnosno monotono opadajuće podnizive niza brojeva koji se nalazi u vektoru koji je zadan kao parametar, redom slijeva nadesno, Recimo, ukoliko vektor zadan kao parametar sadrži niz brojeva 1, 3, 2, 6, 7, 9, 10, 4, 3, 1 i 7, funkcija "`RastuciPodnizovi`" treba da vrati vektor vektôrā čiji je prvi element vektor sa elementima 1 i 3, drugi element je vektor sa elementima 2, 6, 7, 9 i 10, dok je treći (i posljednji) element vektor sa elementima 1 i 7. S druge strane, funkcija

"`OpadajuciPodnizovi`" treba da vrati vektor vektôrā čiji je prvi element vektor sa elementima 3 i 2, dok je drugi (i posljednji) element je vektor sa elementima 10, 4, 3 i 1. Moguće je da izlazni vektor vektôrā bude i prazan (ukoliko ulazni vektor ne sadrži ni jedan monotono rastući odnosno monotono opadajući podniz).

Napisane funkcije demonstrirajte u testnom programu u kojem se sa tastature unose elementi nekog vektora (broj njegovih elemenata se također unosi sa tastature), a koji nakon toga poziva ove funkcije da odredi sve maksimalne monotono rastuće odnosno monotono opadajuće podnizove niza brojeva koji je smješten u ovom vektoru, te da ih ispiše na ekran (svaki podniz u posebnom redu, elementi podnizova trebaju biti međusobno razdvojeni razmacima).

4. U ovom zadatku potrebno je implementirati dvije funkcije za procesiranje teksta, nazvane "`ObrniFraze`" i "`IzmijeniUPigLatin`". Prvo ćemo opisati funkciju "`ObrniFraze`". Njen prvi parametar je neki string (tj. njegov tip je "`std::string`") koji sadrži rečenicu u kojoj treba obrnuti riječi ili fraze. Drugi parametar je vektor stringova (tj. vektor čiji su elementi tipa "`std::string`") koji sadrži popis riječi odnosno fraza koje treba obrnuti. Funkcija treba da kao rezultat vrati modificiranu rečenicu u kojoj je svaka riječ odnosno fraza iz spiska izvrnuta naopačke. Na primjer, ukoliko se kao prvi parametar funkciji proslijedi rečenica "Izasla je prva zadaca iz predmeta Tehnike programiranja, a ovih dana ocekujemo i jos zadaca iz drugih predmeta", a kao drugi parametar vektor koji sadrži stringove "zadaca", "Tehnike programiranja", "drugih predmeta" i "meso od sira", kao rezultat treba da se dobije string koji sadrži rečenicu "Izasla je prva acadaz iz predmeta ajnarimargorp ekinheT, a ovih dana ocekujemo i jos acadaz iz atemderp higurd". Iz primjera je vidljivo i to da ukoliko se neka od fraza ne pronađe nigdje unutar zadane rečenice (poput "meso od sira" u prethodnom primjeru), ne treba da se desi ništa posebno. Postupak zamjene treba precizno da teče ovako. Uzima se redom jedan po jedan string iz spiska fraza. Za svaki takav string traže se njegova pojavljivanja kao podstringa u rečenici, redom slijeva nadesno (razmatra se samo potpuno poklapanje, uključujući i eventualne razmake). Ukoliko se pronađe takav podstring, on se obrće, te nakon obrtanja pomjeramo jedan znak udesno i nastavljamo dalje dok se eventualno ne obrnu sve pjave te riječi/fraze. Nakon toga prelazimo na sljedeću riječ/frazu iz spiska, itd.

   Što se tiče druge funkcije nazvane "`IzmijeniUPigLatin`", ona se tiče igre riječi u engleskom jeziku koja je poznata pod nazivom PigLatin. U toj igri se riječi izmjenjuju po pravilu da se prvo slovo riječi prebaci na kraj, a zatim se na kraj doda slog "ay". Npr., riječ "*pig*" se transformira u "*igpay*" (postoje još neka pravila, ali ovdje ćemo se ograničiti samo na ovo pravilo). Prvi parametar funkcije "`IzmijeniUPigLatin`" je neki string (tipa "`std::string`") koji sadrži rečenicu u kojoj treba zamijeniti neke ili sve riječi prema PigLatin sistemu. Drugi parametar je vektor stringova (tj. vektor čiji su elementi tipa "`std::string`") koji sadrži popis riječi koje treba izmijeniti (svaki element vektora predstavlja jednu riječ). Funkcija treba da kao rezultat vrati modificiranu rečenicu u kojoj je svaka riječ iz spiska transformirana po pravilu PigLatin. Ukoliko se kao drugi parametar funkciji pošalje prazan vektor, onda ona treba izmijeniti sve riječi u rečenici.

   U parametru koji predstavlja spisak riječi, svaki string smije sadržavati samo slova engleskog alfabeta. Ukoliko se u ma kojem stringu nađe bilo šta što ne spada u slova engleskog alfabeta, funkcija treba da baci izuzetak tipa "`domain_error`" uz prateći tekst "Nekorektan izbor riječi".

   Napisanu funkciju demonstrirajte u testnom programu u kojem se traži da se sa tastature unese neka rečenica, a zatim skupina riječi odnosno fraza. Riječi odnosno fraze se unose jedna po jedna, pri čemu se nakon svake riječi odnosno fraze pritišće ENTER. Unos riječi/fraza prestaje kada se samo naprazno pritisne ENTER, bez ičega unesenog. Nakon unosa rečenice, te riječi/fraza, program treba da ispiše unesenu rečenicu sa transformiranim riječima koja se dobija pozivom napisane funkcije "`IzmijeniUPigLatin`" (ovo će raditi ispravno samo ukoliko su unesene samo riječi a ne i fraze, jer će u suprotnom biti bačen izuzetak), a zatim u novom redu rečenicu sa obrnutim riječima/frazama koja se dobija pozivom napisane funkcije "`ObrniFraze`". Ukoliko prva funkcija baci izuzetak, to se ne smije odraziti na nastavak ispisivanja koji se očekuje u zadatku, odnosno i dalje treba ispisati ono što je dobijeno pozivom funkcije "`ObrniFraze`".