Zadaća 1

Ova zadaća nosi ukupno 4 poena, od kojih svaki nosi po 1 poen. Svi zadaci se mogu uraditi na osnovu gradiva sa prva tri predavanja i pretpostavljenog predznanja iz predmeta "Uvod u programiranje" odnosno "Osnove računarstva". Krajnji rok za predaju ove zadaće je nedjelja, 31. III 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 biće kažnjeno nepriznavanjem čitavog* (*inače tačnog*) *zadatka*. Zbot toga, prije slanja dobro provjerite (recimo, pretragom pomoću Ctrl-F) da li su Vam negdje u programu ostale uglaste zagrade!

JOŠ VAŽNIJE: Pri izradi zadataka, najstrožije je zabranjeno koristiti bilo kakve funkcije i objekte koji nisu obrađivani niti na prva 4 predavanja kursa "Tehnike programiranja", niti na kursu "Uvod u programiranje" odnosno "Osnove računarstva" (ovo se posebno odnosi na funkcije iz biblioteke sa zaglavljem "`algorithm`", te na funkcije koje se primjenjuju na objektima tipa "`string`", a koje nisu rađene na predavanjima). *Nepoštovanje ove odredbe biće također kažnjeno nepriznavanjem čitavog* (*inače tačnog*) *zadatka*!!!

1. Godine 1621. francuski matematičar Claude Gaspard Bachet de Méziriac je naslutio da se svaki prirodan broj može napisati kao suma kvadrata najviše četiri prirodna broja, recimo

   $3 = 1^2 + 1^2 + 1^2$
   $25 = 5^2$
   $31 = 5^2 + 2^2 + 1^2 + 1^2$
   $45 = 6^2 + 3^2$
   $310 = 15^2 + 9^2 + 2^2$

   Ovu slutnju napokon je dokazao Joseph Louis Lagrange skoro 150 godina kasnije (1770. godine). Vaš zadatak je da napravite dvije funkcije nazvane "`BrojKvadrata`" i "`RazvrstajPoBrojuKvadrata`". Funkcija "`BrojKvadrata`" prima kao parametar cijeli broj n (tipa "`int`"), a vraća kao rezultat najmanji broj k takav da se n može napisati kao suma od k kvadrata prirodnih brojeva (recimo, za vrijednosti n koje iznose 3, 25, 31, 45 i 310, funkcija redom treba vratiti 3, 1, 4, 2 i 3). Za $n = 0$ funkcija treba vratiti 0 (jer je trivijalna suma od 0 sabiraka jednaka nuli), dok za $n < 0$ funkcija treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Broj mora biti nenegativan". Što se tiče funkcije "`RazvrstajPoBrojuKvadrata`", ona ima dva parametra, od kojih je prvi vektor cijelih brojeva, dok će značenje drugog parametra biti uskoro objašnjeno. Funkcija kao rezultat vraća moderni niz koji se sastoji od 5 vektora cijelih brojeva, a koji predstavlja rezultat razvrstavanja elemenata vektora po minimalnom broju sabiraka koji su potpuni kvadrati a čija suma daje razmatrani element. Konkretnije, i-ti element ovog modernog niza treba da bude vektor koji se sastoji od svih onih elemenata ulaznog vektora za koje funkcija "`BrojKvadrata`" daje i kao rezultat (tj. koji se mogu napisati kao suma od i kvadrata prirodnih brojeva, ali ne i kao suma manje od i kvadrata prirodnih brojeva). Brojevi u tim vektorima koji su dobijeni kao rezultat razvrstavanja trebaju biti u istom relativnom poretku u kakvom su bili u ulaznom vektoru. Što se tiče drugog parametra funkcije "`RazvrstajPoBrojuKvadrata`", on je tipa "`TretmanNegativnih`", pri čemu je "`TretmanNegativnih`" pobrojani tip s ograničenim vidokrugom, koji je deklariran na globalnom nivou pomoću deklaracije

   ```
   enum class TretmanNegativnih {IgnorirajZnak, Odbaci, TretirajKao0, PrijaviGresku};
   ```

   Ovaj parametar određuje kako će se tretirati negativni brojevi prilikom razvrstavanja. Ukoliko on ima vrijednost "`IgnorirajZnak`", negativnim brojevima se prosto ignorira znak, tj. tretiraju se kao da su pozitivni. Ukoliko je vrijednost ovog parametra "`Odbaci`", negativni brojevi se uopće ne razvrstavaju, tj. svi negativni brojevi se odbacuju prilikom razvrstavanja. Ukoliko je vrijednost ovog parametra "`TretirajKao0`", svi negativni brojevi se klasificiraju u klasu 0 (tj. idu u element modernog niza s indeksom 0), odnosno tretiraju se kao da su jednaki nuli. Konačno, ukoliko ovaj parametar ima vrijednost "`PrijaviGresku`", nailazak na negativne brojeve prilikom razvrstavanja

treba da dovede do bacanja izuzetka tipa "`domain_error`" uz prateći tekst "Nije predvidjeno razvrstavanje negativnih brojeva".

Na kraju, potrebno je napisati i mali testni program ("`main`" funkciju) u kojoj ćete demonstrirati napisanu funkciju "`RazvrstajPoBrojuKvadrata`" na sekvenci brojeva koja se unose sa tastature. Brojevi se unose dok se ne unese bilo šta što nije broj (npr. tačka ili riječ "kraj"). Nakon sekvence, program treba da ispiše rezultate razvrstavanja, tako što će se u posebnim redovima ispisati vrijednost k (klasa) koja može biti od 0 do 4, zatim dvotačka, te svi uneseni brojevi koji se mogu napisati kao suma od k sabiraka koji su kvadrati prirodnih brojeva, ali ne i kao suma manje od k sabiraka koji su kvadrati prirodnih brojeva. Redovi su poredani po rastućim vrijednostima k, a ukoliko u klasu koja odgovara broju k nema nijednog broja, taj red se uopće ne prikazuje. Slijedi primjer dijaloga između korisnika i programa:

```
Unesite brojeve (bilo koji ne-broj oznacava kraj): 25 310 0 11 31 3 7 .

Rezultati razvrstavanja po broju kvadrata:
0: 0
1: 25
3: 310 11 3
4: 31 7
```

U slučaju da među unesenim brojevima ima i negativnih brojeva, program treba ispisati tekst "Nije podrzano razvrstavanje negativnih brojeva!", kao u sledećem dijalogu (obratite pažnju da ovaj tekst *nije istovjetan* tekstu bačenom u okviru izuzetka):

```
Unesite brojeve (bilo koji ne-broj oznacava kraj): 1 22 333 -4444 5555 -66666 .

Nije podrzano razvrstavanje negativnih brojeva!
```

NAPOMENA: Jasno je da se od studenata ne očekuje da implementiraju nikakve napredne algoritme zasnovane na teoriji brojeva pomoću kojih se tražena rastava može naći enormno brzo čak i za ogromne brojeve n. Ipak, od studenta se očekujue da bi funkcija "`BrojKvadrata`" trebala raditi razumno brzo (ne duže od nekoliko sekundi) za brojeve reda do $n = 1000000$, što nije teško postići elementatnim logičkim razmišljanjem koje ne zahtijeva nikakve specijalne tehnike.

2. Za potrebe digitalne obrade slika i općenito za potrebe pamćenja slika u računarskoj memoriji, slika se najčešće posmatra kao da je sastavljena od vrlo sitnih nedjeljivih kvadratića, nazvanih *pikseli*. Na taj način, slika se može modelirati kao matrica, pri čemu svaki od elemenata matrice odgovara po jednom pikselu, a sadrži informaciju o svjetlini ili eventualno boji odgovarajućeg piksela (na primjer, svijetlijim pikselima mogu odgovarati veći brojevi, a tamnijim pikselima manji brojevi). Međutim, uslijed grešaka u procesu fotografiranja, zna se desiti da se na slici mogu uočiti izvjesne smetnje, koje se tipično manifestiraju kao mrlje ili slične degradacije u kvalitetu slike. Zbog toga, među najvažnije postupke u digitalnoj obradi slike, spadaju postupci tzv. *filtriranja*, čiji je cilj da se smanji vizualni efekat smetnji.

   Postoji mnogo različitih tehnika filtriranja. Jedna od poznatih tehnika, koja se posebno dobrom pokazala za uklanjanje nečistoća koje se manifestiraju kao vrlo male crne ili bijele mrlje koje su razbacane po slici (poznate kao *šum soli i bibera*, ili engl. *salt-and-pepper noise*) je tzv. medijanski filter (engl. *median filter*). Da bismo objasnili kako radi ova tehnika, moramo prvo objasniti šta je medijan. Ukoliko imamo neku skupinu brojeva, njihov medijan je onaj broj koji se nalazi tačno u sredini ukoliko te brojeve sortiramo po veličini. Recimo, ukoliko imamo skupinu brojeva 35, 28, 23, 14, 44, 17, 28 i 23, njihov medijan je broj 28. Zaista, ovi brojevi sortirani po veličini formiraju spisak 14, 17, 23, 28, 28, 35 i 44, u čijoj se sredini nalazi broj 28. Ukoliko se u skupini nalazi paran broj brojeva, tada je medijan poluzbir dva broja koja se nalaze u sredini sortirane skupine (s obzirom da tada nije jednoznačno koji je tačno element na sredini). Na primjer, za skupinu brojeva 70, 15, 29, 22, 31 i 57, medijan je broj 30. Zaista, ovi brojevi kada ih sortiramo po veličini formiraju spisak 15, 22, 29, 31, 57, 70. U njegovoj sredini su brojevi 29 i 31, a $(29 + 31)/2 = 30$.

   Nakon što smo objasnili šta je medijan, možemo objasniti i tehniku medijanskog filtriranja. Prema ovoj tehnici, oko svakog piksela formira se "prozor" veličine $(2\mathrm{N} + 1) \times (2\mathrm{N} + 1)$, gdje je N tzv. *red filtera* (prirodan broj). Tako, "prozoru" koji odgovara pikselu na poziciji (i, j) pripadaju

svi pikseli kod kojih je prva koordinata u opsegu od $i-\mathrm{N}$ do $i+\mathrm{N}$, a druga koordinata u opsegu od $j-\mathrm{N}$ do $j+\mathrm{N}$. Zatim se vrijednost svakog od piksela zamjenjuje medijanom vrijednosti svih piksela unutar prozora koji pripada tom pikselu. Pri tome, ukoliko se razmatra piksel koji je blizu nekog od ruba matrice, može se desiti da prozor dijelom izađe izvan slike odnosno matrice. Međutim, to nije bitno, s obzirom da se pri računu razmatraju samo oni pikseli koji se zaista nalaze unutar prozora. Drugim riječima, pri računu se uzimaju samo elementi matrice s legalnim koordinatama, tj. koordinatama koje ne izlaze izvan opsega koordinata matrice. Kao primjer, ispod lijevo je prikazana jedna slika-matrica, dok je s desne strane prikazana ista slika-matrica dobijena filtriranjem medijanskim filterom reda $\mathrm{N}=1$ (tako da se filtriranje vrši usrednjavanjem unutar "prozora" veličine 3×3):

$$\begin{pmatrix} 2 & 3 & 3 & 1 & 2 & 3 \\ 1 & 2 & 4 & 3 & 4 & 4 \\ 2 & 3 & 4 & 5 & 3 & 2 \\ 3 & 4 & 3 & 4 & 4 & 2 \\ 1 & 2 & 3 & 3 & 4 & 3 \end{pmatrix} \qquad \begin{pmatrix} 2 & 2.5 & 3 & 3 & 3 & 3.5 \\ 2 & 3 & 3 & 3 & 3 & 3 \\ 2.5 & 3 & 4 & 4 & 4 & 3.5 \\ 2.5 & 3 & 3 & 4 & 3 & 3 \\ 2.5 & 3 & 3 & 3.5 & 3.5 & 3.5 \end{pmatrix}$$

Vaš zadatak je da napravite funkciju "`MedijanskiFilter`" koja kao prvi parametar prima matricu koja se filtrira, organiziranu kao vektor vektôrā realnih brojeva, a kao drugi parametar red filtriranja N (tipa "`int`"). Ukoliko je N negativan, treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Neispravan red filtriranja". Kao rezultat funkcija treba da vrati novu matricu dobijenu kao rezultat filtriranja. Matrica koja se filtrira uopće ne mora imati ispravnu strukturu matrice, odnosno može biti i "grbava". Pri tome se, naravno, u račun uzimaju u obzir samo pikseli koji stvarno postoje unutar "prozora", tj. pikseli kojima odgovaraju legalne koordinate unutar raspona matrice. U svakom slučaju, rezultirajuća matrica imaće isti oblik kao i ulazna matrica.

Napisanu funkciju trebate demonstrirati u testnom programu u kojem se prvo unose dimenzije matrice, zatim i njeni elementi, te željeni red filtriranja, Nakon toga, program treba da ispiše matricu dobijenu kao rezultat filtriranja. Elemente ispišite koristeći prikaz na tačno dvije decimale, pri čemu ćete predvidjeti širinu od ukupno 7 karaktera za ispis svakog elementa. Slijede primjeri dijaloga između korisnika i programa:

```
Unesite broj redova i kolona matrice: 5 6
Unesite elemente matrice:
2 3 3 1 2 3
1 2 4 3 4 4
2 3 4 5 3 2
3 4 3 4 4 2
1 2 3 3 4 3
Unesite red filtriranja: 1

Matrica nakon filtriranja:
   2.00   2.50   3.00   3.00   3.00   3.50
   2.00   3.00   3.00   3.00   3.00   3.00
   2.50   3.00   4.00   4.00   4.00   3.50
   2.50   3.00   3.00   4.00   3.00   3.00
   2.50   3.00   3.00   3.50   3.50   3.50
```

```
Unesite broj redova i kolona matrice: 2 2
Unesite elemente matrice: 1 2 3 4
Unesite red filtriranja: -1

GRESKA: Neispravan red filtriranja!
```

NAPOMENA: U ovom primjeru testnog programa, filtriranje se uvijek vrši nad matricama koje imaju ispravnu strukturu matrice. Međutim, funkcija za filtriranje će biti testirana i s matricama koje nisu takve, tj. s grbavim matricama.

3. Pod *cijelim brojem unutar nekog stringa* podrazumijeva se svaka sekvenca znakova koja eventualno započinje znakom '–' (koji označava da je broj negativan), iza koje slijede znakovi koji mogu biti u opsegu od '0' do '9' (cifre), pri čemu se iza posljednje cifre obavezno nalazi ili kraj stringa, ili neki znak koji nije ni slovo ni cifra, za koji se ne smatra da pripada broju (u skladu s tim, "123xyz" se

ne smatra brojem, jer iza posljednje cifre '3' slijedi znak 'x' koji je slovo). Kao primjer, u stringu "Reperi 2PAC i 50Cent znaju da zbir brojeva 327, 39 i −4162 iznosi −3796! Oni također znaju da 123xy nije broj, mada 456 jeste. Šta se može reći za 0?", brojevi su "327", "39", "−4162", "−3796", "456" i "0".

Napišite funkciju "`BrojeviUStringu`" koja kao parametar prima neki string (tipa "`string`"), a koja kao rezultat vraća vektor koji se sastoji od vrijednosti svih brojeva koji se nalaze u tom stringu, u onom redoslijedu kojem se javljaju u stringu. Kad kažemo "vrijednosti", misli se da vektor ne treba da sadrži nađene brojeve u formi stringova (recimo, za prethodni primjer, on ne treba da sadrži stringove "327", "39", "−4162", "−3796", "456" i "0"), nego u formi odgovarajućih brojčanih vrijednosti cjelobrojnog tipa (tj. treba da sadrži brojčane vrijednosti 327, 39, −4162, −3796, 456 i 0). Kao cjelobrojni tip odaberite tip "**`long long int`**", da maksimalno proširite opseg brojeva za koje je moguće dobiti odgovarajuću brojčanu vrijednost. Međutim, ukoliko je broj takav da odgovarajuća vrijednost ne može stati u tip "**`long long int`**", funkcija treba baciti izuzetak tipa "`range_error`" uz prateći tekst "Prevelik broj" (za ispravan tretman prekoračenja, studentima se savjetuje da prouče dodatak uz predavanja posvećen upravo toj problematici).

Napisanu funkciju treba demonstrirati u kratkom testnom programu koji za string unesen putem tastature ispisuje sve brojeve nađene u njemu, ili odgovarajuću prikladnu poruku. Dijalozi između korisnika i programa trebaju izgledati poput sljedećih:

```
Unesite string:   Reperi 2PAC i 50Cent znaju da zbir brojeva 327, 39 i -4162 iznosi
-3796! Oni također znaju da 123xy nije broj, mada 456 jeste. Šta se može reći za 0?
Brojevi unutar stringa: 327 39 -4162 -3796 456 0
```

```
Unesite string: Rekli smo da 1234567890123456789012345abcde nije broj...
Uneseni string ne sadrzi nijedan broj!
```

```
Unesite string: Broj 12345678901234567890123456 je prevelik i za tip long long int!
PROBLEM: Uneseni string sadrzi prevelik broj!!!
```

4. Vještina pisanja šifrovanih poruka potiče još iz pradavnih vremena. Jedan od najstarijih poznatih metoda šifriranja (koji je vrlo loš metod) je Caesarova šifra, prema kojoj se svako slovo izvorne poruke zamijenjuje sa znakom koji se po abecedi nalazi 3 znaka ispred (uz izuzetak posljednja tri znaka abecede, koji se mijenjaju s prva tri znaka abecede). Mala slova ostaju mala, velika ostaju velika, a znaci koji nisu slova se ne mijenjaju. Tako se, prema Caesarovoj šifri, poruka "Hura, stigla je zadaca iz TP-a!" šifrirano predstavlja kao "Kxud, vwljod mh cdgdfd lc WS-d!". Poopćenje ove šifre je generalizirana Caesarova šifra, koja umjesto pomaka 3 koristi neki proizvoljan pomak k u opsegu od 0 do 25, koji se čuva kao tajna informacija (tzv. ključ). Recimo, generalizirana Caesarova šifra s ključem $k = 17$ za isti primjer poruke daje šifriranu poruku "Ylir, jkzxcr av qrurtr zq KG-r!". Ključ za generaliziranu Caesarovu šifru se često predstavlja kao znak u vidu velikog slova, a ne kao broj, pri čemu 'A' predstavlja 0, 'B', predstavlja 1, 'C' predstavlja 2, itd. Tako, klasična Caesarova šifra koristi ključ 'D', dok za generaliziranu Caesarovu šifru s ključem $k = 17$ možemo reći da koristi ključ 'R'.

   Iako je Generalizirana Caesarova šifra bolja od klasične Caesarove šifre (s obzirom da za uspješno dešifriranje treba znati ključ), lako ju je "razbiti", jer postoji svega 26 mogućih ključeva (možemo isprobati jedan po jedan, dok ne dobijemo nešto smisleno). Zapravo, sve tzv. *monoalfabetske šifre*, kod kojih se isti znak izvorne poruke uvijek zamjenjuje istim znakom u šifriranoj poruci, prilično su lake za "razbijanje". Znatno su sigurnije tzv. *polialfabetske šifre*, kod kojih se isti znak izvorne poruke može mijenjati s više različitih znakova. Jedna od najjednostavnijih takvih šifri je tzv. *Vigenèreova šifra*, koja je bila veoma popularna čak i u vojnim primjenama, sve do kraja prošlog vijeka kada su otkriveni algoritmi za njeno razbijanje. Kod Vigenèreove šifre, ključ nije samo jedno veliko slovo (odnosno broj od 0 do 25), nego string koji se sastoji od više velikih slova. Prvo slovo poruke šifrira se koristeći prvo slovo ključa, na isti način kao kod generalizirane Caesarove šifre. Međutim, drugo slovo poruke šifrira se koristeći drugo slovo ključa, i tako sve dok ne iscrpimo sva slova ključa, nakon čega se za potrebe šifriranja ponovo uzima prvo slovo ključa, itd. Na primjer, ukoliko je ključ "LOPATA", poruka "Hura, stigla je zadaca iz TP-a!" šifrirano se predstavlja kao "Siga, lttuaa ce kosava tn IP-t!" (prvo slovo poruke 'H' šifrira se koristeći ključ 'L', drugo slovo poruke 'u' šifrira se koristeći ključ 'O', i tako sve do šestog slova poruke 't' koje se šifrira koristeći ključ 'A', nakon čega se sedmo slovo poruke 'i' šifrira ponovo koristeći ključ 'L', itd.).

Vaš zadatak je da napravite funkcije "`ViginereSifriranje`" i "`ViginereDesifriranje`". Obje funkcije imaju dva parametra tipa "`string`", od kojih prvi predstavlja poruku koja se šifrira odnosno dešifrira, dok drugi parametar predstavlja ključ. Ključ mora biti string koji se sastoji isključivo od velikih slova. U suprotnom, obje funkcije bacaju izuzetak tipa "`domain_error`" uz prateći tekst "Neispravan kljuc". Ove dvije funkcije obavljaju šifriranje odnosno dešifriranje zadane poruke uz zadani ključ postupkom Vigenèrevog šifriranja. Kao rezultat, funkcije vraćaju odgovarajući string koji predstavlja šifriranu odnosno dešifriranu poruku.

Napisane funkcije iskoristite u demonstracionom programu, koji radi na sljedeći način. Prvo se traži da se s tastature unese ključ. Ukoliko ključ nije ispravan, ispisuje se odgovarajuća poruka i traži se da se unese ponovo ključ, sve dok ključ ne bude ispravan. Zatim se unosi poruka koja se šifrira. Program tada treba da izvrši šifriranje i prikaže šifriranu poruku na ekranu. Nakon toga se od korisnika traži da "pogodi" ključ za potrebe dešifriranja (tj. da ponovo unese ključ). Program će tada pokušati dešifriranje s unesenim ključem. Ukoliko dobijena poruka nije identična izvornoj poruci, prikazuje se pokušaj dešifriranja (tj. neko smeće) i ispisuje poruka "Niste pogodili kljuc, probajte ponovo!". Ista poruka se prikazuje i ukoliko ključ nije legalan. Postupak se ponavlja sve dok korisnik ne "pogodi" ključ, nakon čega se prikazuje korektno dešifrirana poruka i tekst "Bravo, pogodili ste kljuc!" i program prestaje s radom. Primjer dijaloga između korisnika i programa dat je na sljedećoj slici:

```
Unesite kljuc: BABA OD 100 KILA
Neispravan kljuc, ponovite unos:
Unesite kljuc: 3 DEDE OD PO 3 KILE...
Neispravan kljuc, ponovite unos:
Unesite kljuc: LOPATA
Kljuc prihvacen. Unesite poruku: Hura, stigla je zadaca iz TP-a!
Sifrirana poruka: Siga, lttuaa ce kosava tn IP-t!

Pokusajte pogoditi kljuc: Pa budaletino jedna, valjda je onaj isti
Niste pogodili kljuc, probajte ponovo!
Pokusajte pogoditi kljuc: CIGLA
Pokusaj desifriranja: Qaap, lrlopa aw edsynu in GH-n!
Niste pogodili kljuc, probajte ponovo!
Pokusajte pogoditi kljuc: LOPATA
Pokusaj desifriranja: Hura, stigla je zadaca iz TP-a!
Bravo, pogodili ste kljuc!
```