Zadaća 1.

Ova zadaća nosi ukupno 4 poena, pri čemu prva dva zadatka nose po 0,9 poena, a naredna dva 1,1 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, 19. III 2016. do 20.00.

1. Cijeli brojevi sa n cifara koji su jednaki sumi n-tih stepena svih svojih cifara u teoriji brojeva se nazivaju *Armstrongovi* ili *narcisoidni brojevi*. Ovi brojevi su vrlo rijetki (ima ih ukupno svega 88, a svega oko 30-ak ih ima koji su veličine takve da mogu stati u tip "`int`"). Primjeri takvih brojeva su 5 ($5 = 5^1$), 371 ($371 = 3^3 + 7^3 + 1^3$), 8208 ($8208 = 8^4 + 2^4 + 0^4 + 8^4$), itd. Napravite funkciju "`ArmstrongoviBrojevi`" koja ima dva cjelobrojna parametra (tipa "`int`"), nazovimo ih recimo p i q, koja kao rezultat vraća vektor čiji su elementi svi Armstrongovi brojevi u opsegu od p do q uključivo, sortirani u rastući poredak. Na primjer, za $p = 3000000$ i $q = 10000000$, funkcija treba da vrati vektor čiji su elementi 4210818, 9800817 i 9926315 (ovo su Armstrongovi brojevi u navedenom opsegu).

   Napisanu funkciju demonstrirajte u testnom programu koji traži da se sa tastature unesu brojevi p i q, a koji će nakon toga pozvati napisanu funkciju i na osnovu njenog rezultata ispisati sve Armstrongove brojeve u opsegu od p i q uključivo (u rastućem poretku), međusobno razdvojene zarezom (iza posljednjeg elementa ne treba stajati zarez). Ukoliko u zadanom opsegu nema niti jedan Armstrongov broj, treba da se ispiše poruka "Nema Armstrongovih brojeva u traženom opsegu".

   NAPOMENA: Funkcija "`ArmstrongoviBrojevi`" mora se zasnivati na stvarnom testiranju da li je broj Armstrongov ili ne (prema njegovoj definiciji). Recimo, ne dolazi u obzir rješenje da neko u neki niz ili vektor upiše spisak svih Armstrongovih brojeva koji mogu stati u tip "`int`" (a takav spisak se može naći recimo na internetu) i da onda samo "vadi" brojeve iz spiska koji leže u zahtijevanom opsegu. Takvo rješenje će automatski biti bodovano sa 0 poena.

2. Neka je **A** neka matrica formata $m \times n$. Pod $p \times q$ ekspanzijom matrice **A** smatramo novu matricu formata $m\,p \times n\,q$ koja se dobija tako što se svaki element $a_{i,j}$ matrice **A** zamijeni blokom formata $p \times q$ pri čemu su svi elementi tog bloka jednaki $a_{i,j}$. Na primjer, ukoliko matrica **A** glasi

$$\mathbf{A} = \begin{pmatrix} 3 & 5 \\ 4 & -1 \\ 0 & 2 \end{pmatrix}$$

   njena 2×4 ekspanzija glasi

$$\begin{pmatrix} 3 & 3 & 3 & 3 & 5 & 5 & 5 & 5 \\ 3 & 3 & 3 & 3 & 5 & 5 & 5 & 5 \\ 4 & 4 & 4 & 4 & -1 & -1 & -1 & -1 \\ 4 & 4 & 4 & 4 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 2 & 2 & 2 & 2 \\ 0 & 0 & 0 & 0 & 2 & 2 & 2 & 2 \end{pmatrix}$$

   Napišite funkciju "`EkspanzijaMatrice`" sa 3 parametra. Prvi parametar je matrica **A** sa realnim elementima koja je organizirana kao vektor vektōrâ realnih brojeva (tj. vektor čiji su elementi vektori realnih brojeva, i to tipa "`double`"), dok su drugi i treći parametar cijeli brojevi p i q. Funkcija treba da kreira i kao rezultat vrati $p \times q$ ekspanziju matrice **A**. U slučaju da je neki od parametara p ili q nula ili negativan, funkcija treba da baci izuzetak tipa "`domain_error`" uz prateći tekst "Nelegalni parametri". U slučaju da prvi parametar nema ispravnu formu matrice, odnosno ukoliko svi redovi proslijeđenog vektora vektōrâ nemaju isti broj elemenata, treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Nekorektna matrica".

   Napisanu funkciju demonstrirajte u testnom programu u kojem se sa tastature prvo traži unos dva cijela broja m i n, zatim elementi matrice **A** formata $m \times n$, te konačno dva cijela broja p i q. Program nakon toga treba da pozivom napisane funkcije kreira $p \times q$ ekspanziju matrice **A** 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 "Nekorektna matrica" 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. Za neki element matrice, njemu *susjedni elementi* su oni elementi koji se nalaze neposredno lijevo, desno, iznad ili ispod njih, kao i elementi koji se nalaze jedno polje dijagonalno u odnosu na njih (gore lijevo, gore desno, dolje lijevo ili dolje desno). Drugim riječima, ma koji element $a_{i,j}$ može imati najviše 8 susjeda, i to su, polazeći odozgo i gledajući u redoslijedu kazaljke na satu, elementi $a_{i-1,j}$, $a_{i-1,j+1}$, $a_{i,j+1}$, $a_{i+1,j+1}$, $a_{i+1,j}$, $a_{i+1,j-1}$, $a_{i,j-1}$ i $a_{i-1,j-1}$. Naravno, elementi koji se nalaze uz "rubove" matrice neće imati svih 8 susjednih elemenata. Recimo, za elemente koji se nalaze u prvom redu matrice nemaju susjednih elemenata sa gornje strane, odnosno za njih ne postoje elementi $a_{i-1,j}$, $a_{i-1,j-1}$ i $a_{i-1,j+1}$, itd.

   Nakon što smo uveli pojam susjednih elemenata, uvešćemo pojam *vrha matrice*. Za neki element matrice kažemo da je njen vrh ukoliko nijedan njegov susjed nije veći od njega (ukoliko elemente matrice posmatramo kao nadmorske visine nekog terena, geometrijska interpretacija šta je zapravo vrh postaje očigledna). Pošto je ovo pojam lokalnog karaktera, jasno je da ista matrica može imati i više različitih vrhova. Nalaženje nekog vrha matrice može se naravno vršiti "grubom silom", osnosno testiranjem jednog po jednog elementa matrice da li se radi o vrhu ili ne, sve dok se ne pronađe neki vrh. Međutim, obično je mnogo efikasnije za tu svrhu koristiti algoritam iz porodice algoritama koji se obično nazivaju *penjanje uz brdo* (engl. *hill climbing*). Za ovaj konkretan problem, ovaj algoritam bi izgledao ovako. Krenemo od nekog elementa, i ispitamo sve njegove susjede. Ukoliko su svi njegovi susjedi manji ili jednaki od njega, element na kojem se upravo nalazimo je očigledno vrh, i postupak je završen. U suprotnom, pređemo na najveći od svih njegovih susjeda, i ponavljamo dalje postupak od njega, sve dok se vrh ne dostigne.

   Vaš zadatak je da napravite funkciju "`VrhMatrice`" sa 3 parametra, koja nalazi lokaciju nekog od vrhova matrice koristeći strategiju penjanja uz brdo. Prvi parametar je matrica sa cjelobrojnim elementima organizirana kao vektor vektōrâ cijelih brojeva. U slučaju da ovaj parametar nema ispravnu formu matrice, odnosno ukoliko svi redovi proslijeđenog vektora vektōrâ nemaju isti broj elemenata, treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Nekorektna matrica". Drugi i treći parametar predstavljaju respektivno redni broj reda i kolone elementa od kojeg započinje pretraga (pri čemu numeracija redova i kolona počinje od nule). Ukoliko ovi parametri ukazuju na poziciju koja izlazi izvan opsega matrice, treba baciti izuzetak tipa "`range_error`" uz prateći tekst "Nekorektna pocetna pozicija" (bez kvakice na "c", da se izbjegnu problemi sa kodiranjem neengleskih slova). Funkcija treba da kao rezultat vrati kompleksni broj tipa "`std::complex<double>`" čiji realni i imaginarni dio respektivno predstavljaju redni broj reda i kolone gdje je vrh pronađen.

   U slučaju da za neki element postoji više susjeda koji su najveći, vrh koji će biti pronađen može zavisiti od toga na koji smo od tih susjeda prešli. Da bi rezultati bili jednoznačni (što je neophodno radi lakšeg testiranja), susjede testirajte polazeći odozgo i u redoslijedu kazaljke na satu kao što je opisano u uvodnom razmatranju, naravno preskačući pri tome nepostojeće elemente (ukoliko se nalazimo uz rub matrice). U slučaju da više elemenata ima istu najveću vrijednost, treba preći na onaj element na koji naiđemo *prvi* u takvom redoslijedu pretrage.

   Napisanu funkciju demonstrirajte u testnom programu u kojem se sa tastature prvo traži unos dva cijela broja m i n, zatim elementi matrice $\mathbf{A}$ formata $m \times n$, te konačno dva cijela broja p i q. Program nakon toga treba da pozivom napisane funkcije pronađe vrh matrice polazeći od elementa sa indeksima p i q, te da na ekran ispiše na kojoj se poziciji nalazi nađeni vrh i kolika je njegova vrijednost. Obavezno predvidite hvatanje izuzetaka koji bi eventualno mogli biti bačeni iz funkcije. Vrijedi ista napomena kao u prethodnom zadatku.

4. Pretpostavimo da u nekoj rečenici želimo da istaknemo neke riječi ili fraze tako što ćemo ih staviti u zagradu. U ovom zadatku cilj je napisati funkciju "`IstakniFraze`" koji će automatizirati ovaj proces. Prvi parametar ove funkcije je neki string (tj. njegov tip je "`std::string`") koji sadrži rečenicu u kojoj treba istaknuti riječi ili fraze. Drugi parametar je vektor stringova (tj. vektor čiji su elementi tipa "`std::string`") koji sadrži popis riječi odnosno fraza koje treba istaknuti. Funkcija treba da kao rezultat vrati modificiranu rečenicu u kojoj je svaka riječ odnosno fraza iz spiska stavljena u zagrade. 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 (zadaca) iz predmeta (Tehnike programiranja), a ovih dana ocekujemo i jos (zadaca) iz (drugih predmeta)". 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.

   Funkcija "`IstakniFraze`" ne treba biti posebno inteligentna. Princip na kojem ona treba da radi je prosto da se za svaki string koji se nalazi unutar vektora koji je drugi parametar funkcije traže sva njegova eventualna pojavljivanja kao podstringa unutar stringa koji je prvi parametar. Za svako takvo pojavljivanje, oko njega se dodaju zagrade, i to bi u suštini bilo sve. Dakle, po takvom rješenju, "Tehnike programiranja", zatim "Tehnike programiranja" (sa 4 razmaka između dvije riječi), te " Tehnike programiranja " (sa nekoliko razmaka na početku i kraju) tretiraju se kao *različite fraze*. Bolje bi bilo da nije tako, ali bi se bez toga funkcija dodatno zakomplicirala, i studenti to ne trebaju da rade. Također, "zadaca" i "Zadaca" su različite stvari, odnosno poređenje treba biti osjetljivo na razliku imeđu velikih i malih slova.

   Napisanu funkciju demonstrirajte u testnom programu u kojem se traži da se sa tastature unese neka rečenica, a zatim skupina rjieči 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. Po okončanju unosa, program treba da ispiše unesenu rečenicu sa istaknutim riječima/frazama, koja se dobija pozivom napisane funkcije.