Zadaća 2.

Ova zadaća nosi ukupno 4 poena, pri čemu svaki zadatak nosi po 0,8 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 14. IV 2014. (do kraja dana) i ne može se produžiti. Zadaće se predaju putem Zamgera.

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

   $3 = 1^2 + 1^2 + 1^2 + 0^2$
   $31 = 5^2 + 2^2 + 1^2 + 1^2$
   $310 = 17^2 + 4^2 + 2^2 + 1^2$

   Ovu slutnju napokon je dokazao Joseph Louis Lagrange skoro 150 godina kasnije (1770. godine). Vaš zadatak je da napravite funkciju "`SumaCetiriKvadrata`" sa 5 parametara, koja za zadani prirodan broj n nalazi četiri nenegativna cijela broja a, b, c i d takva da je $n^2 = a^2 + b^2 + c^2 + d^2$ i $a \geq b \geq c \geq d$ (zbog komutativnosti sabiranja, ovaj posljednji uvjet je uvijek moguće postići). Prvi parametar funkcije je broj n, a funkcija treba da pronađene vrijednosti za a, b, c i d smjesti respektivno u drugi, treći, četvrti i peti parametar funkcije. U slučaju da je n negativan funkcija treba da baci izuzetak tipa "`domain_error`" koji sadrži prateći tekst "Broj koji se rastavlja mora biti nenegativan". U slučaju da postoji više ispravnih rastava (recimo, broj 45 se može rastaviti kao $45 = 6^2 + 3^2 + 0^2 + 0^2$ ili kao $45 = 6^2 + 2^2 + 2^2 + 1^2$ ili kao $45 = 4^2 + 4^2 + 3^2 + 2^2$ i na još nekoliko načina) potrebno je pronaći *bilo koju* od njih. Napisanu funkciju demonstrirajte na testnom progamu koji za broj unesen sa tastature ispisuje neku od ispravnih rastava broja na sumu kvadrata četiri nenegativna cijela broja.

   NAPOMENA: Od studenata se 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 trebala raditi razumno brzo (ne duže od nekoliko sekundi) za brojeve reda do $n = 1000000$, što nije nimalo teško postići elementatnim logičkim razmišljanjem koji ne zahtijeva nikakve specijalne tehnike.

2. Napišite generičku funkciju "`ZajednickiElementiBlokova`" koja ima pet parametara p_1, p_2, p_3, p_4 i p_5. Parametri p_1 i p_2 omeđuju jedan blok elemenata (p_1 pokazuje na početak bloka a p_2 tačno iza njegovog kraja), dok p_3 i p_4 omeđuju drugi blok elemenata. Funkcija treba da nađe zajedničke elemente ova dva bloka (tj. elemente koji se javljaju i u jednom i u drugom bloku) i da ih upiše u blok na čiji početak pokazuje p_5 (pri tome se podrazumijeva da tamo ima dovoljno prostora da se prihvate nađeni elementi). Ukoliko se isti element javlja više puta kao zajednički, ne treba ga više puta upisivati u odredišni blok. Na primjer, ukoliko prvi blok sadrži elemente 5, 2, 7, 4, 6, 1, 3, 2, 7 i 4 a drugi blok elemente 2, 9, 0, 6, 0, 4, 8, 3, 2 i 5, u odredišni blok treba upisati elemente 5, 2, 4, 6 i 3. Zajednički elementi koji se prepisuju u odredišni blok trebaju da budu u onakvom redoslijedu kakvi su u prvom bloku. Kao rezultat, funkcija treba da vrati pokazivač ili iterator (zavisno da li je p_5 pokazivač ili iterator) koji pokazuje tačno iza kraja bloka u koji sadrži pronađene zajedničke elemente. Parametri p_1 i p_2 trebaju biti istog tipa. Isto vrijedi i za p_3 i p_4. Međutim, tipovi za p_1 i p_3 nisu nužno isti (recimo, p_1 može biti pokazivač, a p_3 iterator). Tip p_5 može biti posve neovisan od tipa ostalih parametara.

   Napisanu funkciju demonstrirajte u testnom programu koji traži da se sa tastature unesu elementi jednog niza jednog deka (elementi niza i deka unose se sa tastature, pri čemu broj elemenata niza neće biti veći od 100) a koji zatim poziva napisanu funkciju da pronađe elemente koji se javljaju i u nizu i u deku i smjesti ih u jedan novi vektor. Potrebno je iskoristiti povratnu vrijednost funkcije da se veličina vektora na kraju namjesti da bude tačno onolika koliko ima nađenih zajedničkih elemenata, nakon čega se nađeni elementi trebaju ispisati na ekran.

   NAPOMENA: Slična funkcija, pod nazivom "`set_intersection`" postoji u biblioteci "algorithm", ali posjeduje ograničenje da blokovi moraju biti sortirani.

3. Napišite generičku funkciju "`KreirajTabeluSabiranja`" sa tri parametra "`tabela`", "`v1`" i "`v2`". Parametri "`v1`" i "`v2`" su vektori proizvoljnog ali istog tipa elemenata, za koje ćemo pretpostavljati da je definirana operacija sabiranja (što uključuje sve brojčane tipove, ali i recimo tip "`string`"). Funkcija treba da dinamički alocira dvodimenzionalni niz (tabelu) sa onliko redova koliko "`v1`" ima elemenata i onoliko kolona koliko "`v2`" ima elemenata, čiji su elementi istog tipa kao i elementi vektora "`v1`" i "`v2`". Nakon toga, funkcija treba da popuni elemente tako kreirane tabele tako da element u *i*–tom redu i *j*-toj koloni bude jednak zbiru *i*–tog elementa vektora "`v1`" i *j*-tog elementa vektora "`v2`" i na kraju, da dodijeli adresu alociranog prostora dvojnom pokazivaču "`tabela`" koji se koristi za pristup elementima tabele (za tu svrhu, formalni parametar "`tabela`" mora biti *referenca na dvojni pokazivač*). Pri tome, koristite postupak *kontinualne alokacije*. Ova funkcija ne vraća nikakvu vrijednost, nego se jedina izlazna informacija iz funkcije prenosi putem formalnog parametra "`tabela`". U slučaju da alokacija ne uspije, funkcija treba kao izuzetak baciti tekst "Kreiranje tabele nije uspjelo", pri čemu ni u kom slučaju ne smije doći do curenja memorije. Napisanu funkciju demonstrirajte u malom testnom programu koji će primijeniti ovu funkciju nad dva vektora kompleksnih brojeva od kojih prvi vektor ima 6 a drugi 4 elementa (elementi se unose putem tastature), a koji će zatim ispisati na ekran elemente kreiranog dvodimenzionalnog niza u tabelarnoj formi. Na kraju, program treba prije završetka da oslobodi memoriju zauzetu kreiranim dvodimenzionalnim nizom (bez obzira što će se to svakako desiti automatski po završetku programa). Oslobađanje treba vršiti pozivom generičke funkcije "`UnistiTabelu`" (koju također trebate napisati), koja kao parametar prima dvojni pokazivač preko kojeg se pristupa tabeli (za razliku od fragmentirane alokacije, u ovom slučaju ovoj funkciji nikakve druge informacije neće biti potrebne).

4. Na predavanjima je obrađeno nekoliko korisnih funkcija iz biblioteke "`algorithm`". Među njima su i funkcije "`find`", "`reverse`" i "`replace_if`" i "`remove_copy_if`". Napišite vlastite verzije ovih funkcija koje ćete nazvati redom "`Nadji`", "`Obrni`", "`ZamijeniUvjetno`" i "`KopirajUzUklanjanjeUvjetno`". Ove funkcije trebaju raditi potpuno istu stvar kao i odgovarajuće bibliotečke funkcije. Sve ove funkcije treba realizirati isključivo korištenjem pokazivačke aritmetike (dakle, indeksiranje i trivijalna simulacija indeksiranja poput pisanja "`*(p+i)`" umjesto "`p[i]`" *nisu dozvoljeni*). Bitno je da sve funkcije moraju biti zasnovane na *potpunoj dedukciji tipova*, tako da ispravno rade i sa pokazivačima i sa iteratorima. Ove funkcije testirajte u glavnom programu koji će:
   - Unijeti sa tastature *n* cijelih brojeva i smjestiti ih u dek, pri čemu se *n* također unosi sa tastature;
   - Ispitati da li se u deku nalazi element 10; ukoliko se nalazi, ispisati poziciju gdje se nalazi i pomnožiti ga sa 3 (u slučaju da ima više takvih elemenata, sve prethodno rečeno odnosi se samo na prvu pojavu tog elementa); ukoliko se ne nalazi, treba ispisati odgovarajuću poruku koja govori o tome;
   - Izvrnuti sve elemente deka (nakon modifikacije obavljene u prethodnoj stavci), tako da prvi element postane posljednji, itd.
   - Zamijeniti sve elemente u deku (nakon prethodne modifikacije) u kojima se nalazi više cifara manjih od 5 nego cifara koje su veće ili jednake od 5 (kao npr. u broju 2371) sa nulom; za tu svrhu će Vam trebati pomoćna funkcija koja testira da li njen cjelobrojni parametar ima više cifara manjih od 5 nego cifara koje su veće ili jednake od 5; takvu funkciju trebate nazvati "`ViseCifaraManjihOd5`";
   - Iskopirati sve elemente tako dobijenog deka koji nisu prosti brojevi u dinamički alocirani niz od *n* cijelih brojeva; pomoćnu funkciju koja testira da li je njen cjelobrojni parametar prost broj ili ne trebate nazvati "`DaLiJeProst`";
   - Ispisati na ekran sve elemente tako formiranog niza (tačnije, samo one elemente koji su zaista kopirani iz deka, pošto njih može biti manje od *n*);
   - Osloboditi memoriju koju je zauzimao dinamički alocirani niz.

   U glavnom programu nije dozvoljeno koristiti petlje (osim za unos i ispis elemenata niza), već sve manipulacije treba ostvariti isključivo pozivima napisanih funkcija. Isto tako, niti jedna funkcija unutar svog tijela ne smije koristiti nikakve dodatne kontejnerske tipove podataka (nizove, vektore, dekove, itd.).

Napomena 1: Pokazivačka aritmetika ne radi nužno ispravno sa dekovima, s obzirom da ne postoji garancija da su susjedni elementi deka zaista i susjedni u memoriji. Stoga se za rad sa dekovima moraju koristiti iteratori. Vodite računa o tome.

Napomena 2: Kao test da li Vam napisane funkcije rade ispravno, Vaš glavni program treba identično raditi ukoliko pozive funkcija "`Nadji`", "`Obrni`", "`ZamijeniUvjetno`" i "`KopirajUzUklanjanjeUvjetno`" zamijenite pozivima odgovarajućih bibliotečkih funkcija.

5. Napravite funkciju "`SortirajPoBrojuDjelilaca`" čiji je prvi parametar pokazivač koji pokazuje na prvi element nekog dinamički alociranog niza cijelih brojeva, dok je drugi parametar broj elemenata u tom nizu. Funkcija treba da sortira razmatrani niz u takav poredak u kojem će brojevi koji imaju više djelilaca dolaziti ispred brojeva sa manje djelilaca. Na primjer, po takvom poretku, broj 16 koji ima 5 djelilaca (1, 2, 4, 8 i 16) treba da se nalazi ispred broja 9 koji ima 3 djelioca (1, 3 i 9), a on opet treba da se nalazi ispred broja 17 koji ima samo 2 djelioca (1 i 17). U slučaju da dva broja imaju isti broj djelilaca, tada manji broj treba da dođe prije većeg broja. Sortiranje treba obaviti uz pomoć bibliotečke funkcije "`sort`" (uz definiranje odgovarajuće funkcije kriterija, koja se treba zvati "`KriterijSortiranja`"). Napisanu funkciju treba demonstrirati u testnom programu koji prvo traži da se sa tastature unese prirodan broj *n*, nakon čega dinamički alocira niz od *n* brojeva i popunjava ih vrijednostima unesenim sa tastature. Program zatim sortira niz u traženi poredak pozivom napisane funkcije. Nakon obavljenog sortiranja, sortirani niz treba ispisati na ekran. Na kraju, program treba da za novi broj unesen sa tastature postupkom binarne pretrage ustanovi da li se nalazi u razmatranom nizu ili ne (podrazumijeva se ispis odgovarajućeg komentara o tome) i da nakon toga obriše dinamički alocirani niz. Za tu svrhu, napisaćete pomoćnu funkciju nazvanu "`PretraziBinarno`", koja prima iste parametre kao i funkcija "`SortirajPoBrojuDjelilaca`" samo uz dodatni treći parametar koji predstavlja broj koji se traži. Funkcija treba vratiti logičku vrijednost "tačno" ili "netačno" u ovisnosti od toga da li se traženi broj nalazi ili ne nalazi u nizu. Funkcija "`PretraziBinarno`" treba unutar sebe pozivati odgovarajuću bibliotečku funkciju za binarnu pretragu. Dobro obratite pažnju da niz nije sortiran u uobičajenom poretku – funkcija za obavljanje binarne pretrage to mora uzeti u obzir!