# 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 petak 13. IV 2012. (do kraja dana) i ne može se produžiti. Zadaće se predaju putem Zamgera.***

***Pitanje:*** ***Da li je moguće produžiti rok za predaju zadaće do nedjelje, 15. IV 2012. s obzirom da je ispit u ponedjeljak, 16. IV 2012?***

***Odgovor:*** ***Ne.***

***Pitanje:*** ***Zašto?***

***Odgovor:*** ***Baš zato što je ispit u ponedjeljak, 16. IV 2012. Zadaća se ne radi dan pred ispit. A ni dva dana pred ispit. Posljednja dva dana pred ispit trebaju se iskoristiti na bolji način nego izradom zadaće.***

***Pitanje:*** ***A ne možete odgoditi ni zbog činjenice da se petak 13. smatra baksuznim danom?***

***Odgovor:*** ***Ne može... Niste valjda toliko sujevjerni!?***

***Pitanje:*** ***Da li je ovo konačan stav?***

***Odgovor:*** ***Da (evo i jednog afirmativnog odgovora).***

***VAŽNA NAPOMENA: Postoji mogućnost da će se neki zadaci testirati automatskim testnim skriptama. Da bi takvo testiranje ispravno radilo, sve funkcije se MORAJU ZVATI TAČNO ONAKO KAKO JE SPECIFICIRANO i moraju primati TAČNO ONAKVE PARAMETRE KAKO JE SPECIFICIRANO. U suprotnom će testna skripta odbaciti zadatak kao neispravan!!!***

1. Napišite generičku funkciju "`OdstraniZadaneElemente`" koja ima dva parametra "`v`" i "`v1`". Oba parametra su vektori proizvoljnog ali istog tipa elemenata (tj. tip elemenata u oba vektora je isti) za koje se pretpostavlja da se mogu porediti. Funkcija treba da iz vektora "`v`" odstrani sve elemente koji se nalaze i u vektoru "`v1`", zadržavajući ostale elemente u istom poretku kakvi su bili prije odstranjivanja. Funkcija ne vraća nikakvu vrijednost, već samo utiče na elemente parametra "`v`" (koji pri tome, naravno, može promijeniti svoju veličinu). Na primjer, ako prije poziva funkcije vektor "`v`" sadrži redom elemente 3, 8, 5, 6, 1, 4, 9, 7, 2, 2, 6, 4, 9, 1, 4, 8, 3, 6 i 5, a vektor "`v1`" elemente 4, 0, 3, 4 i 2, nakon poziva funkcije vektor "`v`" treba da sadrži redom elemente 8, 5, 6, 1, 9, 7, 6, 9, 1, 8, 6 i 5. Pri tome, funkcija ne smije u svom radu kreirati i koristiti nikakve druge vektore ili nizove osim samih parametara "`v`" i "`v1`" (tj. nije dozvoljeno koristiti nikakve pomoćne vektore). Također, nije dozvoljeno koristiti nikakve bibliotečke funkcije čija upotreba nije bila demonstrirana na predavanjima (poput funkcije "`erase`" primijenjene na vektore). Napisanu funkciju demonstrirajte u testnom programu koji će iz spiska kompleksnih brojeva koji se unose sa tastature (te kompleksne brojeve treba čuvati u vektoru čiji su elementi kompleksni brojevi) odstraniti sve kompleksne brojeve sa drugog spiska kompleksnih brojeva koji se također unosi putem tastature i ispisati sve kompleksne brojeve iz prvog spiska nakon obavljenog odstranjivanja.

2. Napišite program koji od korisnika traži da unese dimenzije pravougaone matrice $n$ i $m$ a zatim da unese dvije matrice formata $n \times m$ (pretpostavite da su elementi matrica realni brojevi). Program nakon toga treba da ispiše produkt dvije unesene matrice. Program treba da bude zasnovan na skupini napisanih funkcija za rad sa matricama, koje radi univerzalnosti treba izvesti kao generičke funkcije (bez obzira što će se u ovom programu raditi samo sa matricama čiji su elementi realni brojevi). U programu treba da se nalaze sljedeće funkcije: "`KreirajMatricu`", "`UnesiMatricu`", "`PomnoziMatrice`", "`IspisiMatricu`" i "`UnistiMatricu`". Funkcija "`KreirajMatricu`" prima kao parametre dvojni pokazivač koji služi za pristup dinamički kreiranoj matrici (u nastavku ćemo ovaj pokazivač prosto zvati *dinamička matrica*), kao i dimenzije matrice $n$ i $m$. Funkcija treba da alocira prostor za matricu formata $n \times m$ i dodijeli adresu alociranog prostora pokazivaču koji se koristi za pristup njenim elementima. Funkcija "`UnesiMatricu`" popunjava matricu elementima unesenim sa tastature, a prima kao parametre

→

dinamičku matricu i dimenzije *n* i *m*. Funkcija "`PomnoziMatrice`" prima kao parametre dvije dinamičke matrice i njihove dimenzije *n1* i *m1*, odnosni *n2* i *m2*. Ova funkcija treba da *kreira* novu dinamičku matricu (pozivom funkcije "`KreirajMatricu`"), da je popuni proizvodom dvije dinamičke matrice koje su joj proslijeđene kao parametri, i da vrati kao rezultat dvojni pokazivač koji služi za pristup elementima novokreirane matrice. U slučaju da matrice nisu saglasne za množenje, funkcija treba da baci izuzetak. Funkcija "`IspisiMatricu`" kao parametre prima dinamičku matricu, dimenzije *n* i *m* kao i željenu širinu ispisa, a ispisuje elemente matrice na ekran, pri čemu se svaki element matrice ispisuje u skladu sa zadanom širinom ispisa. Konačno, funkcija "`UnistiMatricu`" unistava dinamički kreiranu matricu koja joj se prosljeđuje kao parametar (zajedno sa dimenzijama *n* i *m*). Funkcija "`KreirajMatricu`" treba da baci izuzetak u slučaju da kreiranje ne uspije. Pri tome, ova funkcija mora da vodi računa da u slučaju da dođe do bacanja izuzetka "počisti iza sebe" sve uspjele alokacije, tako da ne dođe do curenja memorije. Eventualno bačene izuzetak treba hvatati u glavnom programu. Obavezno testirajte slučaj kada alokacija ne uspijeva (unosom prevelikih brojeva *n* i/ili *m*).

3. Na predavanjima je obrađeno nekoliko korisnih funkcija iz biblioteke "`algorithm`". Među njima su i funkcije "`min_element`", "`reverse`" i "`find_if`" i "`replace_copy_if`". Napišite vlastite verzije ovih funkcije nazvane "`Najmanji`", "`Izvrni`", "`NadjiUvjetno`" i "`KopirajUzZamjenuUvjetno`", koje rade potpuno istu stvar kao i odgovarajuće bibliotečke funkcije. Funkcije treba realizirati isključivo korištenjem pokazivačke aritmetike. 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;
   - Umanjiti najmanji broj u deku za 1 (ukoliko ima više najmanjih brojeva, treba umanjiti samo prvi od njih);
   - Ispisati koji je prvi element u deku (nakon modifikacije obavljene u prethodnoj stavci) koji ima barem jednu parnu cifru u sebi (npr. broj 3527) i na kojoj se poziciji u deku nalazi (ukoliko takvog elementa nema, treba ispisati odgovarajuću poruku koja govori o tome);
   - Izvrnuti sve elemente deka, tako da prvi element postane posljednji, itd.
   - Iskopirati sve elemente tako izvrnutog deka u dinamički alocirani niz od *n* cijelih brojeva, uz zamjenu onih elemenata koji su prosti brojevi nulama;
   - Ispisati na ekran sve elemente tako formiranog niza;
   - 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: Iako na prvi pogled tako ne izgleda, potpuna dedukcija može prilično usložniti implementaciju nekih od traženih funkcija. Jednu od njih čak nećete moći implementirati bez da napišete jednu pomoćnu funkciju (također generičku) koju ćete pozivati iz nje (što naravno nije zabranjeno). Sami ćete uvidjeti zašto.

   Napomena 2: 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 3: Kao test da li Vam napisane funkcije rade ispravno, Vaš glavni program treba identično raditi ukoliko pozive funkcija "`Najmanji`", "`Izvrni`", "`NadjiUvjetno`" i "`KopirajUzZamjenuUvjetno`" zamijenite pozivima odgovarajućih bibliotečkih funkcija.

4. Napravite funkciju "`SortirajRecenice`" koja kao parametar prima pokazivač koji pokazuje na prvi element nekog dinamički alociranog niza stringova (preciznije niza čiji su elementi tipa "`string`") kao i broj elemenata u tom nizu. Funkcija treba da sortira razmatrani niz stringova u

takav poredak u kojem će se duže rečenice nalaziti uvijek ispred kraćih. U slučaju da dvije rečenice imaju istu dužinu, tada rečenica koja dolazi prije po abecednom poretku treba da dođe ispred rečenice koja dolazi kasnije po abecednom poretku (pri tome ne treba praviti razliku između malih i velikih slova). Sortiranje treba obaviti uz pomoć bibliotečke funkcije "`sort`" (uz definiranje odgovarajuće funkcije kriterija). 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* stringova i popunjava ih sa *n* rečenica unesenih 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 novu riječ unesenu 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. Binarnu pretragu treba obaviti uz pomoć odgovarajuće bibliotečke funkcije. Dobro obratite pažnju da niz nije sortiran u uobičajenom poretku – funkcija za obavljanje binarne pretrage to mora uzeti u obzir!

5. Napišite program koji će Vas uvjeriti koliko je bolje koristiti bibliotečku funkciju "`sort`" od ručnog sortiranja. U programu ćete prvo definirati dvije funkcije "`GenerirajNiz`" i "`SortirajRucno`". Funkcija "`GenerirajNiz`" treba da ima jedan cjelobrojni parametar "`n`" i ona dinamički alocira niz od "`n`" elemenata popunjen slučajnim realnim brojevima iz opsega od 0 do 1 i vraća kao rezultat pokazivač na prvi element tako kreiranog niza. Za generiranje slučajnih brojeva koristite funkciju "`rand`" bez parametara iz biblioteke "`cstdlib`" koja kao rezultat vraća slučajan cijeli broj u opsegu od 0 do neke velike vrijednosti nazvane "`RAND_MAX`" a koja je ovisna od implementacije, tako da ćete dijeljenjem vrijednosti koje vrati funkcija "`rand`" sa ovom vrijednošću sigurno dobiti broj u opsegu od 0 do 1 (oprez: čuvajte se cjelobrojnog dijeljenja). Funkcija "`SortirajRucno`" ima dva parametra koji redom predstavljaju pokazivač na prvi element i pokazivač iza posljednjeg elementa bloka koji se sortira (identično kao bibliotečka funkcija "`sort`"). Ova funkcija vrši sortiranje bloka u opadajući poredak koristeći neki od jednostavnih postupaka za sortiranje koji su Vam poznati (npr. BubbleSort, SelectionSort, itd.). Funkcija bi trebala da bude generička, tako da može sortirati u opadajući poredak niz elemenata proizvoljnog tipa uz pretpostavku da se oni mogu međusobno porediti (mada ćete je u ovom programu koristiti samo za sortiranje niza realnih brojeva). Napisane funkcije ćete iskoristiti u testnom programu koji prvo definira dvije cjelobrojne konstante "`n1`" i "`n2`" (čije ćete tačne vrijednosti odrediti kasnije) a koji zatim dinamički kreira dva niza sa respektivno "`n1`" i "`n2`" elemenata. Nakon toga, prvi niz treba sortirati pozivom funkcije "`SortirajRucno`", a drugi pozivom bibliotečke funkcije "`sort`" (uz odgovarajuću funkciju kriterija, s obzirom da se traži sortiranje u opadajućem poretku). Testiranje započnite od malih vrijednosti "`n1`" i "`n2`", a zatim povećavajte ove vrijednosti sve dok trajanje svakog od sortiranja bude trajalo približno 10 sekundi. Uporedite ove vrijednosti i sami izvucite zaključak.

   Napomena: Finalna verzija programa ne treba da traži nikakav unos podataka sa tastature (konstante "`n1`" i "`n2`" trebaju da budu hard-kodirane, tj.fiksno upisane u program)!